Second Quantization with SNEG

A practical guide for solid-state theory

SNEG documentation project

Edition for SNEG 2.2

Licensed under Creative Commons Attribution 4.0 International.

Preface

An electron hopping between two orbitals is described by a bilinear. The same bilinear becomes a current when differentiated with respect to a bond phase, and an exchange interaction when iterated through a high-energy charge state. Keeping the operators until the last step makes these connections visible. This book uses the Wolfram Language package SNEG [1] to do the operator algebra and small-system calculations that connect those descriptions.

Part I follows the construction of a calculation from declarations to matrix elements. Part II can be read by subject: each worked calculation specifies its Hamiltonian, units and basis. Examples headed as case studies start an independent kernel session with SNEG loaded; evaluate the inputs within each study in sequence. Input labels restart accordingly. A final `TableForm` or `MatrixForm` formats an answer for display; store the underlying list or matrix before applying it. Part III gives the calling conventions of the documented symbols, with a page index at the back. Exercises and their solutions use the same conventions as the chapters. Energies include additive constants unless stated otherwise; $\hbar = k_{\mathrm{B}} = 1$.

Contents

Preface **ii**

I Tutorial **1**

1 Getting started **2**
1.1 Installation and a first session 2
1.2 From an operator to a spectrum 3
1.3 Reading operator output 4

2 Operators and their algebra **5**
2.1 What a declaration fixes 5
2.2 Products, scalars and Hermitian conjugation 5
2.3 Reference states 6
2.4 Bosons, Majoranas and spin operators 7
2.5 Transitions between many-body states 7

3 Building Hamiltonians **9**
3.1 One bond at a time 9
3.2 Local interactions and constraints 10
3.3 Pairing and the Nambu representation 11

4 Symbolic sums and momentum space **12**
4.1 An index belongs to its sum 12
4.2 Momentum transfer and Fourier convention 12
4.3 Periodic clusters and boundary twists 13

5 States, bases and exact diagonalization **15**
5.1 Fix the mode order first 15
5.2 Block labels versus degeneracies 15

5.3 Matrix elements, eigenstates and thermal traces 18

6 Expectation values and Wick's theorem **20**
6.1 An average needs a state 20
6.2 Direct and exchange contractions 20
6.3 Finite-temperature and anomalous averages 21

7 Working efficiently **23**
7.1 Expand at the step that needs it 23
7.2 How large is the matrix? 24
7.3 Tests before a long run 24

II Applications in solid-state theory **26**

8 Electrons in crystals **27**
8.1 From the gas to a lattice 27
8.2 Orbital basis, valleys and topology 28
8.3 Flux per plaquette 31

9 Lattice vibrations and electron–phonon coupling **33**
9.1 Quantizing a displacement 33
9.2 Density and bond couplings 33
9.3 Displaced oscillator and induced attraction 34
9.4 Virtual phonons and coupled modes 36

10 Correlated electrons **38**
10.1 One orbital: charge, spin and particle–hole symmetry 38
10.2 Exact solution of the Hubbard dimer 41
10.3 Eliminating virtual charge excitations 43
10.4 Measuring a charge gap on a finite cluster 47
10.5 The atomic propagator and Hubbard-I 49
10.6 Can a static field account for the singlet energy? 52
10.7 Local constraints: Hubbard operators and auxiliary particles . 53
10.8 A three-orbital atom: Hund's rule from the spectrum 55
Exercises 56

11 Superconductivity **58**
11.1 From a Cooper pair to the gap equation 58
11.2 Nambu doubling and bound states 59
11.3 Odd-parity pairing and end modes 61
11.4 Pair pseudospin on a bipartite lattice 62

12 Magnetism **63**
12.1 Exchange on a cluster 63
12.2 Spin waves and their regime of validity 64
12.3 One dimension: strings and boundaries 65
12.4 Itinerant and single-ion magnetism 66

13 Quantum impurities and nanostructures **69**
13.1 A local level embedded in a band 69
13.2 From valence fluctuations to Kondo exchange 70
13.3 Wilson chain and NRG Ljubljana 71
13.4 Charge stability and superconducting islands 72

14 Fermi liquids, Green's functions and response **73**
14.1 Resolvent or thermal trace? 73
14.2 High-frequency checks 74
14.3 A static limit is a prescription 74
14.4 Quasiparticle parameters and available phase space 76

III Function reference **77**

15 Operators, algorithms and interfaces **78**
15.1 Declaring Operators and Parameters 78
15.2 Operator Algebra 93
15.3 Building Hamiltonians 105
15.4 Spin and Isospin 125
15.5 Sums, Lattices, and Momentum Space 148
15.6 States, Bases, and Matrices 170
15.7 Expectation Values and Wick's Theorem 220
15.8 Quadratic Hamiltonians and Mean-Field Theory 225

15.9 Effective Hamiltonians 237
15.10 Equations of Motion and Green's Functions 240
15.11 Output and Conversion 243
15.12 Extending SNEG 246
15.13 Additional interfaces 255

A Conventions at a glance **260**

B Extending the operator algebra **262**
B.1 Normal order as a reduction rule 262
B.2 Scalars and linear maps 263
B.3 Compatibility and tests 263
B.4 Interfaces used by existing models 264

C Solutions to the exercises **267**

Bibliography **275**

Index of functions **277**

Part I

Tutorial

1 Getting started

If two electrons of opposite spin occupy a localized orbital, removing the up electron must leave a down electron. Changing the order in which the two creators were applied changes the sign of the resulting ket, but cannot change its occupation. This distinction between an amplitude and a probability is a good first test of an operator calculation.

1.1 Installation and a first session

Install the release paclet with `PacletInstall` in Mathematica 12.1 or later. The English help pages require Mathematica 14.0 or later. The installation command, which contacts the release server, is:

```
PacletInstall[
 "https://github.com/rokzitko/sneg/releases/latest/download/SNEG.paclet"
    ]
```

Load the package *before* entering a definition using its symbols. The lowercase loader also works on existing SNEG notebooks. In the Documentation Center open `paclet:SNEG/guide/SNEG`.

In[1]:=
```
Needs["sneg`"];
$SnegVersion
```

Out[2]= 2.2.0

Choose a new operator name and declare its algebra. In the following input `d[]` means one spin-degenerate orbital, not an annihilator. The latter needs both a type and a spin index.

In[3]:=
```
snegfermionoperators[d];
u = d[AN, UP]; v = d[AN, DO];
TableForm[{{antikomutator[u, conj[u]], antikomutator[u, conj[v]]},
           {antikomutator[v, conj[u]], antikomutator[v, conj[v]]}},
  TableHeadings -> {{"up", "down"}, {"up", "down"}}]
```

Out[5]=

	up	down
up	1	0
down	0	1

The unit diagonal expresses Pauli exclusion for a single spin-orbital; the off-diagonal zeros express orthogonality. The rule says nothing about the occupation of the orbital. That information belongs to a state or density matrix.

1.2 From an operator to a spectrum

Give the level an energy ϵ and assign an extra cost U when both spins are present. The algebra itself is independent of their values. A separate line declares the parameters real so that conjugation later has the intended meaning.

In[6]:=
```
snegrealconstants[ε, U];
h = ε number[d[]] + U hubbard[d[]];
blocks = makematricesbzvc[h, qsbasisvc[{d[]}]];
TableForm[{#[[1, 1]], #[[1, 2]], #[[2, 1, 1]]} & /@ blocks,
  TableHeadings -> {None, {"Q", "Spin", "Energy"}}]
```

Out[9]=

Q	Spin	Energy
-1	0	0
0	$\frac{1}{2}$	ϵ
1	0	$U + 2\epsilon$

There are four states but only three rows. The middle row represents both projections of a spin 1/2. The label $Q = N - 1$ counts electrons relative to one electron per site. If $U > 0$, changing ϵ from positive to negative first makes the spin doublet the ground state. Only below $\epsilon = -U$ is it favourable to add the second electron. Chapter 10 revisits these two addition thresholds as poles of a Green's function.

1.3 Reading operator output

A dagger distinguishes creation from annihilation in the formatted output. To inspect the actual expression, request `InputForm`; use `FullForm` to see every head and argument. Both are views. For instance, the occupation polynomial represented by `hubbard[d[]]` is unchanged if its displayed product acquires a minus sign from putting all creators on the left. Use `snegtex` when the result is needed as a formula in a paper. A useful final step before export is to compare a matrix element, rather than a particular ordering of symbols.

Case study: Local moment as an occupation measurement

For a single orbital, $(n_\uparrow - n_\downarrow)^2 = n - 2n_\uparrow n_\downarrow$, so the local moment is maximal for single occupancy:

In[1]:=
```
snegfermionoperators[d];
Expand[pow[number[d[], UP] - number[d[], DO], 2] - (number[d[]] - 2
    nc[number[d[], UP], number[d[], DO]])]
```

Out[2]= 0

Exercise 1.1 (*Parity spectroscopy*). At $\epsilon = -U/3$, $U > 0$, determine the addition and removal costs from the ground multiplet. Which excited level would be invisible in a calculation restricted to the same charge sector?

2 Operators and their algebra

2.1 What a declaration fixes

The algebra of an operator is specified by its head, not its printed letter. For fermions a mode is the tuple of indices *after* the type `CR` or `AN`. Changing any member of this tuple gives an orthogonal mode. For spinful electrons the last index denotes spin; an orbital, site, band or channel index belongs before it. Thus $c^\dagger_{2a\uparrow} c_{2b\uparrow}$ transfers charge between orbitals at fixed site and spin. SNEG keeps the same mode ordering under conjugation and reverses the order of factors in a product.

Case study: Changing the local spin multiplicity

Operators with spin $3/2$ have four spin components, $\sigma = -3/2, \ldots, 3/2$:

In[1]:=
```
snegfermionoperators[{f, 3/2}];
number[f[]]
```

Out[2]= $f^\dagger_{-\frac{3}{2}} f_{-\frac{3}{2}} + f^\dagger_{-\frac{1}{2}} f_{-\frac{1}{2}} + f^\dagger_{\frac{1}{2}} f_{\frac{1}{2}} + f^\dagger_{\frac{3}{2}} f_{\frac{3}{2}}$

2.2 Products, scalars and Hermitian conjugation

With $\{c_i, c_j^\dagger\} = \delta_{ij}$, transposing neighbouring factors generates a lower-degree term as well as a sign. Ordinary Wolfram Language multiplication sorts its arguments and cannot encode this operation. Use `nc[a,b,c]` and `pow[a,n]` for operator products and their powers; use ordinary multiplication for commuting coefficients. Declare real constants with `snegrealconstants` and complex amplitudes with `snegcomplexconstants`. If a hopping amplitude can carry flux, its conjugate is a different coefficient. Real functions and real-valued

indices are separate declarations: $\epsilon(k)$ may be real even if its functional form is unknown.

In[1]:=
```
snegfermionoperators[c];
snegcomplexconstants[v];
a = v nc[c[CR, 1, UP], c[AN, 2, UP]];
h = a + conj[a];
Expand[{h - conj[h], komutator[h, number[c[1]] + number[c[2]]]}]
```

Out[5]= $\{0, 0\}$

The two zeros answer different questions: the reverse hop is present with the conjugate amplitude, and every term removes as many particles as it creates. Hermitian pairing would pass the first test but generally fail the second.

2.3 Reference states

In `EMPTY` order an annihilator kills the vacuum. In `SEA` order the filled modes have negative energy or momentum index relative to the chosen Fermi point. A creator below the surface therefore kills the sea. At the Fermi point one must specify the state or resolve the degeneracy; changing the order alone does not determine its physical occupation. `NONE` preserves products within a family and is useful when explicitly comparing before and after a contraction. Changing the ordering cannot undo simplifications of an already evaluated product.

Case study: A particle and a hole above the reference sea

A particle-hole excitation $|\mathrm{ph}\rangle = c^\dagger_{2\uparrow} c_{-1\uparrow}|\mathrm{FS}\rangle$ is normalized, has one electron in the level 2, and one electron (spin down) in the level -1:

In[1]:=
```
snegfermionoperators[d];
ordering[d] = SEA;
ph = nc[d[CR, 2, UP], d[AN, -1, UP], VACUUM];
{nc[conj[ph], ph], nc[conj[ph], number[d[2]], ph], nc[conj[ph],
    number[d[-1]], ph]}
```

Out[4]= $\{1, 1, 1\}$

2.4 Bosons, Majoranas and spin operators

For a boson $[b, b^\dagger] = 1$ replaces the fermion anticommutator. A self-adjoint SNEG Majorana obeys $\{\gamma_i, \gamma_j\} = \delta_{ij}$: its square is 1/2, rather than one. A spin of size s, declared with `snegspinoperators`, acts in dimension $2s + 1$ and commutes with operators at other sites. Local products can be reduced using $S^+S^- = s(s+1) - (S^z)^2 + S^z$; a spin raising operator is not a canonical boson. A Grassmann coefficient is yet another object: it anticommutes with odd fermionic quantities but supplies no creation–annihilation contraction.

Case study: Does a local fermion pair behave as a boson?

The pair operator $b = c_\downarrow c_\uparrow$ is not a boson: $[b, b^\dagger] = 1 - n$:

In[1]:=
```
snegfermionoperators[d];
b = nc[d[AN, DO], d[AN, UP]];
komutator[b, conj[b]]
```

Out[3]= $1 - d^\dagger_\downarrow d_\downarrow - d^\dagger_\uparrow d_\uparrow$

2.5 Transitions between many-body states

Hubbard operators obey $X^{ab}X^{cd} = \delta_{bc}X^{ad}$ on the same site. On different sites the parity of the transition matters: moving an electron from one site to another still carries a Fermi sign even if individual orbital operators have been eliminated.

Case study: Grading local transitions

Fermionic X-operators on different sites anticommute; bosonic ones commute:

In[1]:=
```
snegxoperators[X, "HubbardAtom"];
{nc[X[2, "UP", "0"], X[1, "0", "UP"]], nc[X[2, "UP", "DO"], X[1, "0",
     "UP"]]}
```

Out[2]= $\{-X_1^{0UP}X_2^{UP0}, X_1^{0UP}X_2^{UPDO}\}$

Symbolic site indexes; $\{X_i^{0\uparrow}, X_j^{\uparrow 0}\} = \delta_{ij}(X^{00} + X^{\uparrow\uparrow})$:

```
In[3]:= antikomutator[X[i, "0", "UP"], X[j, "UP", "0"]]
```

Out[3]= $(X_j^{00} + X_j^{UPUP})\delta_{i,j}$

Exercise 2.1 (*Majorana normalization*). Write $f = (\gamma_1 + i\gamma_2)/\sqrt{2}$. Determine f^2, $\{f, f^\dagger\}$ and $f^\dagger f$ using the SNEG normalization.

3 Building Hamiltonians

3.1 One bond at a time

For a real amplitude t, `-t hop[c[i],c[j]]` includes both directions and both spins. The first term in a sum over distinct unordered bonds should therefore carry $-t$, not $-t/2$. On a finite open chain the last bond ends at L; on a periodic chain it joins L to 1. The distinction is measurable already in the one-electron spectrum.

Case study: Standing waves and open ends

Open tight-binding chain with $N = 5$ sites. The single-particle levels are $-2t\cos(\pi m/(N+1))$, $m = 1, \ldots, N$:

In[1]:=
```
snegfermionoperators[c];
snegrealconstants[t];
H = -t Sum[hop[c[j], c[j + 1]], {j, 4}];
h = matrixrepresentationop[H, Table[c[CR, j, UP], {j, 5}]]
```

Out[4]=
$$\begin{pmatrix} 0 & -t & 0 & 0 & 0 \\ -t & 0 & -t & 0 & 0 \\ 0 & -t & 0 & -t & 0 \\ 0 & 0 & -t & 0 & -t \\ 0 & 0 & 0 & -t & 0 \end{pmatrix}$$

In[5]:=
```
Sort[Eigenvalues[h]] == Sort[Table[-2 t Cos[Pi m/6], {m, 5}]]
```

Out[5]= True

The open-chain wave function has nodes just outside the lattice at sites 0 and $L + 1$. Its momentum denominator is consequently $L + 1$. Periodic boundary conditions instead identify sites separated by L. For complex hopping write

$vc_i^\dagger c_j + \text{h.c.}$ explicitly or use `matrixhop`; multiplying the entire Hermitian bond by complex v does not insert a Peierls phase.

Case study: Testing a bond by its current

Continuity equation on a three-site chain $H = -t\,(\text{hop}_{12} + \text{hop}_{23})$. The rate of change of the occupancy of the middle site, $\dot{n}_2 = i[H, n_2]$ ($\hbar = 1$), equals the particle current into site 2 from both neighbors, $J_{1\to2} + J_{3\to2}$ with $J_{j\to i} = -t$ `current[c[i], c[j]]`:

In[1]:=
```
snegfermionoperators[c];
snegrealconstants[t];
H = -t (hop[c[1], c[2]] + hop[c[2], c[3]]);
Expand[I komutator[H, number[c[2]]] - (-t current[c[2], c[1]] - t
    current[c[2], c[3]])]
```

Out[4]= 0

3.2 Local interactions and constraints

The number operator, the double-occupation projector and the spin Casimir give three distinct probes of a local shell. On one spinful orbital $D = n_\uparrow n_\downarrow$ is a projector, not n^2: $n^2 = n + 2D$. A penalty $\lambda(n-1)^2$ lifts both even charge states, whereas UD acts only on the doublon. Only at fixed total charge can one freely trade a uniform term proportional to n for a constant.

Case study: When does the orbital carry spin?

The square of the spin of a level is $\mathbf{S}^2 = \frac{3}{4}P_1$:

In[1]:=
```
Expand[spinss[d[]] - 3/4 projector1[d[]]]
```

Out[1]=
$$\frac{3\text{number}(d(),0)\text{number}(d(),1)}{4} + \frac{3\text{number}(d(),1)\text{number}(d(),0)}{4} - \frac{3}{4}\text{number}(d(),0) - \frac{3}{4}\text{number}(d(),1) + \text{spinss}(d())$$

3.3 Pairing and the Nambu representation

A source for a singlet pair is invariant under spin rotations but not under a change of electronic phase with its coefficient held fixed. The Hermitian conjugate of a pair *reverses* the order of the two annihilators. A sign error at this step can turn real eigenvalues into imaginary ones rather than simply changing the sign of the gap.

Case study: A complex pairing amplitude

Construction of a Hermitian operator: the BCS pairing term $\Delta c^\dagger_\uparrow c^\dagger_\downarrow + \text{h.c.}$:

In[1]:=
```
snegfermionoperators[c];
snegcomplexconstantsΔ[];
pair = Δ nc[c[CR, UP], c[CR, DO]];
pair + conj[pair]
```

Out[4]= $\Delta(-c^\dagger_\downarrow c^\dagger_\uparrow) + c_\downarrow c_\uparrow \Delta^*$

In a doubled basis $\Psi = (a, a^\dagger)^T$ one writes $H = \tfrac{1}{2}\Psi^\dagger M \Psi + C$. The factor $1/2$ compensates for doubling; the constant resulting from commuting a through $a^\dagger$ must be kept when comparing energies. For a short open Hubbard chain `hamiltonian` is a convenient model constructor; its hopping parameter enters with a plus sign. The explicit bond sum is preferable for a ring or spatially varying coefficients.

Exercise 3.1 (*Counting bonds*). Compare a two-site molecule with a periodic chain of length two whose unit-cell sum contains a forward hop and its conjugate at each site. Find both one-particle spectra for real $t > 0$. Why is dividing by two not a general remedy for rings with $L > 2$?

4 Symbolic sums and momentum space

4.1 An index belongs to its sum

In $\sum_k a_k^\dagger a_k$ the value of k has no meaning outside the sum. Multiplying two copies requires two independent summation indices, even if the same letter was used to write both. Restricting them to be equal would throw away every term coupling two different levels. The symbolic head `sum` preserves this distinction; `Sum` with numerical limits instead enumerates a cluster.

Case study: Nested summations

Nested sums are merged:

In[1]:=
```
snegfreeindexes[i, j];
snegrealfunctions[g];
sum[sum[g[i, j], {i}], {j}]
```

Out[3]=
$$\sum_{i,j} g(i,j)$$

4.2 Momentum transfer and Fourier convention

For N_s Bravais cells take

$$c_R = N_s^{-1/2} \sum_k e^{ik\cdot R} c_k, \qquad c_k = N_s^{-1/2} \sum_R e^{-ik\cdot R} c_R.$$

One independent sum supplies one factor N_s when a lattice phase is constant. A density bilinear brings $1/N_s$ but its site sum brings N_s back; a local quartic

vertex retains $1/N_s$ and three independent momenta. Declare lattice and momentum indices with the *same* lattice size. Keep `blochphase` symbolic when combining deltas; convert to exponentials only for an explicit dispersion or geometry.

Case study: Following the transfer through a density operator

Momentum conservation. The density operator $\rho_q = \sum_k c^\dagger_{k+q\uparrow} c_{k\uparrow}$ transfers the momentum q: $[c_{p\uparrow}, \rho_q] = c_{p-q,\uparrow}$:

In[1]:=
```
snegfermionoperators[ck];
snegmomentumindexes[{k, p, q}, Ns];
sumSimplifyKD[komutator[ck[AN, p, UP], sum[nc[ck[CR, k + q, UP], ck[
    AN, k, UP]], {k}]]]
```

Out[3]= $\mathrm{ck}_{p-q\uparrow}$

4.3 Periodic clusters and boundary twists

In one dimension $c_{n+L} = e^{i\theta} c_n$ selects the grid $k_m = (2\pi m + \theta)/L$. Bilinears involve differences of two momenta: the twists cancel in a transferred momentum. Thus it is incorrect to shift *every* momentum in a momentum-conservation delta by θ/L. `clusterhop` inserts the phase only when a real-space hop crosses the boundary; `fourierrules` gives the explicit finite-dimensional change of basis. In a periodic untwisted cluster `sumCluster` folds indices modulo each lattice dimension.

Case study: Flux sensitivity of the spectrum

Persistent current. The single-particle energies of a ring threaded by the flux θ are $-2t\cos((2\pi m + \theta)/L)$; the ground-state energy of one electron depends on the flux:

In[1]:=
```
snegspinlessfermionoperators[f];
snegrealconstants[t, theta];
H = clusterhop[-t, f, f, 1, {3}, "Twist" -> theta];
h = Table[braketop[f[CR, i, 0], H, f[CR, j, 0]], {i, 0, 2}, {j, 0,
    2}];
Simplify[Eigenvalues[h /. {t -> 1, theta -> Pi/2}] - Table[-2 Cos[(2
    Pi m + Pi/2)/3], {m, 0, 2}]]
```

Out[5]= $\{0, 0, 0\}$

Exercise 4.1 (*Discrete unitarity*). Show that a common twist does not change the inner product of two columns of the finite Fourier matrix. Test $L = 3$, $\theta = \pi/2$ and explain why a reversed sign in just the inverse transform fails.

5 States, bases and exact diagonalization

5.1 Fix the mode order first

Let $|n_1 \dots n_M\rangle = (a_1^\dagger)^{n_1} \cdots (a_M^\dagger)^{n_M}|0\rangle$. An annihilator acting on slot j picks up $(-1)^{\sum_{i<j} n_i}$ before it reaches that slot. Neither the binary occupation vector nor the site labels alone determine this sign: the associated `BASIS` is part of the representation. Keep the mode order when moving a matrix into another numerical code. An operator-form state instead retains the creators explicitly and can be inspected without decoding bit positions.

In[1]:=
```
snegfermionoperators[d];
makebasis[{d[]}];
{BASIS, ap[d[AN, DO], vc[1, 1]]}
```

Out[3]= $\{\{d_\uparrow^\dagger, d_\downarrow^\dagger\}, -\mathrm{vc}(1,0)\}$

5.2 Block labels versus degeneracies

Use charge and spin projection if $[H, N] = [H, S^z] = 0$. Use total spin only if H commutes with all components. A transverse field can break S^z; a pairing term breaks N but may preserve spin and fermion parity. Projecting onto a non-invariant block loses couplings rather than proving that they vanish. One diagnostic for a projector P is $(1 - P)HP = 0$. A second is to compare the reconstructed full spectrum for a cluster small enough that symmetries are unnecessary.

Case study: Counting states in a highest-weight basis

Three spinful orbitals have 64 many-body states. List the number of independent highest-weight states at each charge and spin; the last column restores the omitted spin projections:

In[1]:=
```
snegfermionoperators[c];
b = qsbasisvc[c /@ Range[3]];
dims = {Sequence @@ First[#], Length[Last[#]], (2 #[[1,2]]+1) Length[
    Last[#]]} & /@ b;
TableForm[dims, TableHeadings -> {None,{"Q","S","Multiplets","States
    "}}]
```

Out[4]=

Q	S	Multiplets	States
−3	0	1	1
−2	$\frac{1}{2}$	3	6
−1	0	6	6
−1	1	3	9
0	$\frac{1}{2}$	8	16
0	$\frac{3}{2}$	1	4
1	0	6	6
1	1	3	9
2	$\frac{1}{2}$	3	6
3	0	1	1

The sum rule counts copies of each representation as well as its dimension:

In[5]:=
```
Total[dims[[All,-1]]] == 4^3
```

Out[5]= True

Case study: Separating mirror-even and mirror-odd states

Reflect a three-site molecule about its centre. In the one-electron sector the two end orbitals form even and odd combinations; the central orbital is even. Construct the occupation-vector basis with definite reflection parity from the operator basis and read its dimensions:

In[1]:=
```
snegfermionoperators[c];
sites = c /@ Range[3];
b = transformtoLRvc[qsbasis[sites],sites,vacuum[]];
one = Select[b, #[[1,1]] == -2 &];
TableForm[{Sequence @@ First[#],Length[Last[#]]} & /@ one,
 TableHeadings -> {None,{"Q","S","Parity","Multiplets"}}]
```

Out[5]=

Q	S	Parity	Multiplets
-2	$\frac{1}{2}$	-1	1
-2	$\frac{1}{2}$	1	2

An impurity attached only to the central site hybridizes with the even channel. A calculation in a single parity block would miss an odd bound state altogether; it is the observable's symmetry, not just that of the Hamiltonian, that decides which blocks are needed. Reflection exchanges sites, not just occupation numbers. A permutation of creators must be reordered into the reference order. This can supply a minus sign even when the *pattern* of occupied sites is invariant. Likewise particle–hole conjugation maps Q to $-Q$ and interchanges local double occupation and the vacuum. On a bipartite lattice at particle–hole symmetry one can use a continuous charge isospin instead of just pairing states of opposite charge.

Case study: Spin and charge multiplets on a bipartite dimer

The half-filled bipartite Hubbard dimer has independent spin and charge SU(2) symmetries. Use the particle-hole symmetric interaction and opposite phases for the pair-lowering operators at the two ends of the bond. Check the commutator before reducing the basis:

In[1]:=
```
snegfermionoperators[c];
snegrealconstants[t,U];
sites = {c[1],c[2]};
tm = isospinminus[c[1],0] + isospinminus[c[2],1];
h = -t hop[c[1],c[2]] + U Total[isozsq[sites]]/2;
Expand[komutator[h,tm]]
```

Out[6]= 0

Each row below is one multiplet, not a single state. For example the $I = 1$,

$S = 0$ row contains the vacuum, the fully occupied state and a half-filled singlet; all have the same energy with this energy origin:

In[7]:=
```
bz = transformQStoIS[qsbasisvc[sites],tm];
mm = makematricesbzvc[h,bz];
rows = Join @@ Table[{Sequence @@ First[blockdata],#} & /@
    Eigenvalues[Last[blockdata]], {blockdata,mm}];
TableForm[rows, TableHeadings -> {None,{"Isospin","Spin","Energy"}}]
```

Out[10]=

Isospin	Spin	Energy
1	0	U
$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}(U-2t)$
$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}(2t+U)$
0	1	0
0	0	$\frac{1}{2}\left(U-\sqrt{16t^2+U^2}\right)$
0	0	$\frac{1}{2}\left(\sqrt{16t^2+U^2}+U\right)$

Restoring the missing projections recovers all 16 Fock states:

In[11]:=
```
Total[(2 #[[1]]+1)(2 #[[2]]+1) & /@ rows]
```

Out[11]= 16

5.3 Matrix elements, eigenstates and thermal traces

In each invariant sector the matrix element is $\langle i|H|j\rangle$, with the ket indexed by the column. Wolfram Language `Eigensystem` returns eigenvectors as a list of vectors. For a numerical normalized eigenvector v compute an expectation as $v^\dagger A v$, not $v^T A v$. For an operator changing quantum numbers, such as $c^\dagger$, a matrix with both indices in the same sector may be identically zero. Use `makeallmatricesbzvc` to obtain the rectangular blocks between sectors.

Case study: A partition function from reduced spin blocks

Thermodynamics of the Hubbard dimer: in the (Q, S) basis the partition function is $Z = \sum_{Q,S} (2S+1) \sum_i e^{-\beta E_i^{(Q,S)}}$. At $\beta = 0$, Z is the dimension of the Fock space, 16:

In[1]:=
```
snegfermionoperators[c];
snegrealconstants[t, U];
mats = makematricesbzvc[-t hop[c[1], c[2]] + U Total[hubbard[{c[1], c
    [2]}]], qsbasisvc[{c[1], c[2]}]];
Z[beta_] := Total[(2 #[[1, 2]] + 1) Total[Exp[-beta Eigenvalues
    [#[[2]]]]] & /@ mats];
{Z[0], Z[1] /. {t -> 1, U -> 4} // N}
```

Out[5]= $\{16, 12.6017\}$

Exercise 5.1 (*Weight of a multiplet*). The two-spin Heisenberg dimer has one singlet at $-3J/4$ and one triplet at $J/4$. Derive $\langle \mathbf{S}_1 \cdot \mathbf{S}_2 \rangle_T$ and check both temperature limits for $J > 0$.

6 Expectation values and Wick's theorem

6.1 An average needs a state

Normal ordering is an algebraic operation, whereas contraction depends on the reference state. For a number-conserving Slater determinant the only nonzero two-point functions in its natural-orbital basis are $\langle c_i^\dagger c_j \rangle = \delta_{ij} n_i$ and $\langle c_i c_j^\dagger \rangle = \delta_{ij}(1 - n_i)$. In a paired state the anomalous contractions need not vanish. Specifying the occupations alone is then insufficient. In an interacting thermal state even the complete set of two-point functions need not determine a four-point function.

6.2 Direct and exchange contractions

For $c_i^\dagger c_j^\dagger c_l c_k$ in a determinant the contraction of i with k and j with l has positive sign. Pairing i with l and j with k requires one extra exchange. For $i = j$ these terms cancel exactly. The same minus sign makes the exchange energy negative for a repulsive density interaction. The case study keeps the spin labels distinct until the contraction has been taken.

Case study: The exchange energy of a filled sea

Hartree-Fock energy. In a Slater determinant, the expectation value of the two-particle operator splits into the direct (Hartree) and the exchange (Fock) terms, $\langle c_{1\sigma}^\dagger c_{2\sigma'}^\dagger c_{3\sigma'} c_{4\sigma} \rangle = n_1 n_2 (\delta_{14}\delta_{23} - \delta_{\sigma\sigma'}\delta_{13}\delta_{24})$:

In[1]:=
```
snegfermionoperators[c];
ordering[c] = SEA;
snegfreeindexes[k1, k2, k3, k4, s1, s2];
SimplifyKD[vevwick[nc[c[CR, k1, s1], c[CR, k2, s2], c[AN, k3, s2], c[
    AN, k4, s1]]]]
```

Out[4]= $\theta(-\mathrm{k1})\theta(-\mathrm{k2})(\delta_{\mathrm{k1,k4}}\delta_{\mathrm{k2,k3}} - \delta_{\mathrm{k1,k3}}\delta_{\mathrm{k2,k4}}\delta_{\mathrm{s1,s2}})$

For the electron gas with $H_{\mathrm{int}} = \frac{1}{2V}\sum_{kpq\sigma\sigma'} V_q c^\dagger_{k+q\,\sigma} c^\dagger_{p-q\,\sigma'} c_{p\sigma'} c_{k\sigma}$, the direct term requires $q = 0$ (it cancels against the positive background), and the exchange term gives $E_{\mathrm{x}} = -\frac{1}{2V}\sum_{kq\sigma} V_q n_k n_{k+q}$. In an oscillator vacuum each interchange contributes a plus sign. The number of pairings now appears explicitly in higher moments.

Case study: Gaussian moments of a displacement

Moments of the harmonic-oscillator coordinate $x = (a + a^\dagger)/\sqrt{2}$ in the ground state, $\langle x^{2n}\rangle = (2n-1)!!\,\langle x^2\rangle^n$ (Gaussian distribution):

In[1]:=
```
snegbosonoperators[a];
x = (a[AN] + a[CR])/Sqrt[2];
Table[vevwick[pow[x, n]], {n, 2, 6, 2}]
```

Out[3]= $\{\frac{1}{2}, \frac{3}{4}, \frac{15}{8}\}$

6.3 Finite-temperature and anomalous averages

For a quadratic fermion Hamiltonian the exact thermal state remains Gaussian. Diagonalizing its Nambu matrix gives a covariance matrix; Wick contractions with that matrix recover the expectation of any polynomial in the fermions. This route avoids a trace over 2^M many-body states for M spin-orbitals. A number projector on the full state generally destroys this simple Gaussian structure.

Case study: Checking the anomalous contraction

For on-site singlet pairing, the occupation, the pair amplitude, and the double occupancy. By Wick's theorem, $\langle n_\uparrow n_\downarrow \rangle = \langle n_\uparrow \rangle \langle n_\downarrow \rangle + |\langle c_\downarrow c_\uparrow \rangle|^2 = v^4 + u^2 v^2 = v^2$:

In[1]:=
```
snegfermionoperators[c];
H = 4/5 number[c[1]] - 3/5 (nc[c[CR, 1, UP], c[CR, 1, DO]] + nc[c[AN,
     1, DO], c[AN, 1, UP]]);
gaussianaverage[{number[c[1], UP], nc[c[AN, 1, DO], c[AN, 1, UP]],
    hubbard[c[1]]}, H, {c[1]}, 0]
```

Out[3]= $\{0.1, 0.3, 0.1\}$

Exercise 6.1 (*Wick versus a fixed-charge state*). Compare an equal mixture of the two one-electron states of an atom with a free spin-degenerate level at half occupation per spin. Show that all normal one-body averages coincide but the two double occupations differ. What information has been lost?

7 Working efficiently

7.1 Expand at the step that needs it

For a four-fermion interaction written as UD the compact notation hides the number of terms obtained after a transformation of orbitals. Replacing every creator by a sum of M creators can produce M^4 terms before momentum selection. Apply Kronecker deltas while the indices to be eliminated are still visible. A good intermediate observable is often a commutator or a scalar matrix element rather than the expanded Hamiltonian itself.

Case study: Separating expansion from collection of sums

`sum` does not distribute over sums automatically; `sumExpand` and `sumCollect` convert between the two forms:

In[1]:=
```
snegfreeindexes[k];
snegrealfunctions[f, g];
s = sum[f[k] + g[k], {k}];
{s == sum[f[k], {k}] + sum[g[k], {k}], sumExpand[s], sumCollect[
    sumExpand[s]]}
```

Out[4]=
$$\{\sum_k (f(k)+g(k)) = \sum_k f(k) + \sum_k g(k), \sum_k f(k) + \sum_k g(k), \sum_k (f(k)+g(k))\}$$

Case study: Recovering the physical spin coupling

An interaction written as a linear combination of density and spin-flip monomials may hide a rotationally invariant exchange. Ask `SnegSimplify` to identify local building blocks; inspect the heads of the held constructors without expanding them again:

In[1]:=
```
snegfermionoperators[c];
snegrealconstants[J,U];
raw = Expand[J spinspin[c[1],c[2]] + U hubbard[c[1]]];
compact = SnegSimplify[raw];
Cases[compact, HoldPattern[HoldForm[h_[___]]] :> SymbolName[h],
    Infinity]
```

Out[5]= {hubbard, spinspin}

Recognition is useful only if it preserves the expression. Release the display wrappers and test the operator identity rather than comparing printed strings:

In[6]:=
```
Expand[ReleaseHold[compact] - raw]
```

Out[6]= 0

7.2 How large is the matrix?

For L spinful orbitals the unrestricted basis has 4^L states, so a dense matrix contains 16^L entries. At $L = 8$ a dense complex double-precision array already requires 64 GiB just for its elements. A sparse hopping Hamiltonian instead connects each Fock state to at most a number of others proportional to the number of bonds. Symmetry reduction and sparse storage address different parts of this cost. For a spectral line one often needs only the initial sector and the sectors connected by the probe, rather than all transitions.

7.3 Tests before a long run

Start with one bond. Check Hermiticity, a conserved quantity and one matrix element whose sign is known. Next check a limiting value: does the hopping spectrum reduce to on-site levels at $t = 0$? Only then increase the cluster. For degenerate eigenvalues compare spectral projectors or subspace weights; two diagonalizers may choose different orthonormal vectors inside the same eigenspace.

A short export example checks that the energy symbol and spin label survive conversion to LaTeX:

In[1]:=
```
snegfermionoperators[c];
snegrealconstants[ε];
tex = snegtex[ε number[c[], UP]];
StringContainsQ[tex, "\\epsilon"] && StringContainsQ[tex, "\\uparrow
    "]
```

Out[4]= True

Save the mode list with any exported numerical matrix and record whether additive constants and chemical-potential terms are included. A sorted eigenvalue list alone cannot detect a reversed site order.

Exercise 7.1 (*An independent matrix test*). Find the trace of $U \sum_i D_i$ over the unrestricted Fock space of L spinful orbitals. Would checking only that trace detect an incorrect sign in the hopping? Give a second trace that would distinguish missing bonds.

Part II

Applications in solid-state theory

8 Electrons in crystals

8.1 From the gas to a lattice

For the three-dimensional unpolarized electron gas let $4\pi(r_s a_0)^3/3 = 1/n$, so that $k_F = (3\pi^2 n)^{1/3}$. The Fermi sphere gives the kinetic energy per electron $3\hbar^2 k_F^2/(10m)$. To see the different density scaling of exchange, integrate $4\pi e_*^2/q^2$ over two occupied spheres of equal spin, where $e_*^2 = e^2/(4\pi\epsilon_0)$. At fixed separation q the overlap volume is $\pi(4k_F + q)(2k_F - q)^2/12$ for $q \le 2k_F$ and zero otherwise. The radial measure cancels the interaction denominator. Dividing by the density and including the exchange factor $-1/2$ gives $E_x/N = -3e_*^2 k_F/(4\pi)$. The neutralizing background removes the Hartree energy. Thus kinetic and exchange terms scale respectively as r_s^{-2} and r_s^{-1}; their competition alone does not give the exact correlation energy.

On a Bravais lattice with hopping $-t$ on each nearest-neighbour bond, $\epsilon_k = -2t\sum_{\alpha=1}^{d}\cos k_\alpha$ for the chain, square and simple cubic lattices with unit spacing. Bandwidth $4dt$ does not determine the density of states: at a stationary point its shape also depends on the signs of the local curvatures. A 2D saddle gives the logarithmic van Hove singularity at zero energy on the square lattice; the 1D band edge instead has an inverse-square-root divergence.

Case study: A square-lattice Bloch matrix

Square lattice of spinless fermions:

In[1]:=
```
snegspinlessfermionoperators[f, fk];
snegrealconstants[t];
sneglatticeindexes[{R}, Ns];
snegmomentumindexes[{k}, Ns];
sneglatticevectors[a1, a2];
H = -t sum[hop[f[R], f[R + a1]] + hop[f[R], f[R + a2]], {R}];
h = blochhamiltonian[fourier[H, f -> fk, R -> k], {fk}, k];
Simplify[ExpToTrig[blochphaseexp[h, {k -> {kx, ky}, a1 -> {1, 0}, a2
    -> {0, 1}}]]]
```

Out[8]= $\begin{pmatrix} -2t(\cos(\mathrm{kx}) + \cos(\mathrm{ky})) \end{pmatrix}$

8.2 Orbital basis, valleys and topology

Take one A and one B orbital per cell. In the SSH chain the off-diagonal matrix element in a cell-periodic convention is $q(k) = v + we^{-ik}$. Its path is a circle about v. With chiral symmetry and a fixed cell and termination, this path encloses the origin if $|w| > |v|$; one cannot change that winding without crossing $q = 0$. The spectrum $\pm|q|$ is insensitive to the orientation of the path, but the phase of an eigenvector is not. At $v = w$ the gap closes at $k = \pi$ for real positive hoppings.

Case study: Where the dimerized-chain gap closes

Su-Schrieffer-Heeger chain. The intracell hopping v and the intercell hopping w give the off-diagonal Bloch matrix with $h_{AB}(k) = v + we^{-ik}$, $h_{BA}(k) = h_{AB}(k)^*$, and the bands $\pm\sqrt{v^2 + w^2 + 2vw\cos k}$:

In[1]:=
```
snegspinlessfermionoperators[f, fk];
snegrealconstants[v, w];
sneglatticeindexes[{R}, Ns];
snegmomentumindexes[{k}, Ns];
Hssh = fourier[sum[v hop[f[R, "A"], f[R, "B"]] + w hop[f[R + 1, "A"],
     f[R, "B"]], {R}], f -> fk, R -> k];
h = blochhamiltonian[Hssh, {fk["A"], fk["B"]}, k]
```

Out[6]= $\begin{pmatrix} 0 & w\,\mathrm{blochphase}(-k,1) + v \\ w\,\mathrm{blochphase}(k,1) + v & 0 \end{pmatrix}$

In[7]:=
```
Simplify[Eigenvalues[ExpToTrig[blochphaseexp[h]]]]
```

Out[7]= $\{-\sqrt{2vw\cos(k)+v^2+w^2}, \sqrt{2vw\cos(k)+v^2+w^2}\}$

8.2.1 Measuring a winding without choosing eigenvector phases

An eigensolver may multiply its eigenvector by a different phase at every momentum. Direct finite differences of these vectors are then meaningless. In a two-band chiral model with no diagonal terms use the phase of $q(k) = h_{AB}(k)$ instead. Along a closed mesh add $\arg[q(k_{j+1})/q(k_j)]$, taking each step on the principal branch. The unit cell and ordered pair of orbitals must stay fixed. For the orientation $q = v + we^{-ik}$ the nontrivial integer is negative. The occupied-band Berry phase modulo 2π is π times this winding modulo 2π in the cell-periodic convention.

In[1]:=
```
winding[v_,w_,n_] := Module[{q = N[v + w Exp[-2 Pi I Range[0,n-1]/n
    ]]},
  Round[Total[Arg[RotateLeft[q]/q]]/(2 Pi)]];
TableForm[Table[{v,1,winding[v,1,40],winding[v,1,80]}, {v,{1/2,2}}],
 TableHeadings -> {None,{"v","w","40 points","80 points"}}]
```

Out[2]=

v	w	40 points	80 points
$\frac{1}{2}$	1	-1	-1
2	1	0	0

At $v = w$ the mesh includes $q(\pi) = 0$, so the winding is undefined; do not repair division by zero by adding a small imaginary shift. A phase step can cross the branch cut on a coarse mesh even short of a closing. Refining the mesh or following the circle analytically distinguishes this sampling error from a transition.

8.2.2 A second translation vector

In graphene the two independent translation vectors supply a second phase. At a valley the three complex hopping amplitudes sum to zero, and linearizing their sum gives a Dirac cone. A staggered sublattice potential gives equal mass signs in the two valleys for a fixed spin; intrinsic Kane–Mele spin-orbit coupling gives opposite signs and reverses them for the other spin. Time

reversal therefore allows opposite spin-resolved Chern numbers with zero total Chern number in the spin-conserving limit.

Case study: Destructive interference at a Dirac point

Each A orbital has B neighbours in cells R, $R - a_1$, and $R - a_2$. Extract the structure factor $-h_{AB}/t$; the other off-diagonal entry is its conjugate:

In[1]:=
```
snegspinlessfermionoperators[f, fk];
snegrealconstants[t];
sneglatticeindexes[{R}, Ns]; snegmomentumindexes[{k}, Ns];
sneglatticevectors[a1, a2];
hg = -t sum[hop[f[R,"A"], f[R,"B"]] + hop[f[R,"A"], f[R-a1,"B"]] +
  hop[f[R,"A"], f[R-a2,"B"]], {R}];
h = blochhamiltonian[fourier[hg,f -> fk,R -> k], {fk["A"],fk["B"]}, k
    ];
Expand[-h[[1,2]]/t]
```

Out[7]= $1 + \text{blochphase}(-k, \text{a1}) + \text{blochphase}(-k, \text{a2})$

Evaluate the spectrum at the zone centre, a valley and an edge midpoint, in units of $t > 0$. The bands touch at the valley. The Bravais lattice constant is one:

In[8]:=
```
bands[p_] := Sort[Eigenvalues[Simplify[blochphaseexp[h,
  {k -> p, a1 -> {1,0}, a2 -> {1/2,Sqrt[3]/2}}] /. t -> 1]]];
TableForm[bands /@ {{0,0},{4 Pi/3,0},{0,2 Pi/Sqrt[3]}},
 TableHeadings -> {{"Centre","Valley","Midpoint"},{"Lower / t","Upper
    / t"}}]
```

Out[9]=

	Lower / t	Upper / t
Centre	−3	3
Valley	0	0
Midpoint	−1	1

Case study: Which spin component splits the band?

Rashba chain. With $T = -t\,\mathbf{1} + i\alpha\sigma_y$ on every bond, the Bloch Hamiltonian is $h(k) = -2t\cos k - 2\alpha \sin k\,\sigma_y$, and the bands $-2t\cos k \mp 2\alpha \sin k$ are spin split:

In[1]:=
```
snegfermionoperators[c, ck];
snegrealconstants[t, alpha];
sneglatticeindexes[{R}, Ns];
snegmomentumindexes[{k}, Ns];
H = sum[matrixhop[-t IdentityMatrix[2] + I alpha PauliY, c[R], c[R +
    1]], {R}];
hk = ExpToTrig[blochphaseexp[blochhamiltonian[fourier[H, c -> ck, R
    -> k], {ck}, k]]]
```

Out[6]=
$$\begin{pmatrix} -2t\cos(k) & 2i\alpha\sin(k) \\ -2i\alpha\sin(k) & -2t\cos(k) \end{pmatrix}$$

In[7]:=
```
Eigenvalues[hk]
```

Out[7]= $\{2(-\alpha\sin(k) - t\cos(k)), 2(\alpha\sin(k) - t\cos(k))\}$

8.3 Flux per plaquette

In Landau gauge the forward y hop at coordinate x carries $e^{2\pi i\alpha x}$ for flux α in units of a flux quantum per plaquette. For $\alpha = p/q$ in lowest terms a magnetic cell contains q sites. At fixed k_y the diagonal entries of the Harper matrix are $-2t\cos(k_y + 2\pi px/q)$; nearest x neighbours are coupled by $-t$ with a boundary twist e^{iqk_x}. Here energies are in units of t; at the centre of the magnetic Brillouin zone:

In[3]:=
```
harper[q_, ky_] := Table[-2 Cos[ky + 2 Pi i/q] KroneckerDelta[i, j] -
  Boole[Mod[i-j, q] == 1] - Boole[Mod[j-i, q] == 1],
  {i, 0, q-1}, {j, 0, q-1}];
harper[3, 0] // MatrixForm
```

Out[4]=
$$\begin{pmatrix} -2 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{pmatrix}$$

The oriented product of link phases around a plaquette is gauge invariant; individual bond phases need not be. The same distinction matters for current operators derived by differentiating the hopping with respect to vector potential [2].

Exercise 8.1 (*A magnetic cell*). For flux 1/3 find the three energies at $k_x = k_y = 0$. Check their sum and the sum of their squares without using the eigenvectors.

9 Lattice vibrations and electron–phonon coupling

9.1 Quantizing a displacement

For a monatomic chain with spring constant K and mass M, $\omega_q = 2\sqrt{K/M}\,|\sin(qa/2)|$. For $\omega_q > 0$ the normal coordinate is proportional to $(b_q + b_{-q}^\dagger)/\sqrt{2M\omega_q}$; the adjoint momentum involves their difference. Pairing q with $-q$ keeps the real-space displacement Hermitian. The centre-of-mass translation at $q = 0$ is a free coordinate, not an oscillator for which one may substitute $\omega = 0$ into this normalization.

9.2 Density and bond couplings

A local Holstein term $g(n_i - n_0)(b_i + b_i^\dagger)$ describes a change of site energy. On a bond, modulation of the hopping instead gives $\alpha(u_{i+1} - u_i)\sum_\sigma (c_{i\sigma}^\dagger c_{i+1,\sigma} + \text{h.c.})$. Uniform displacement drops out of the bond coupling. Fourier transforming it supplies a factor $e^{iqa} - 1$ that a local optical-mode vertex does not have. In either case a phonon carries the momentum transferred between electron states.

Case study: Checking the momentum of the emitted phonon

Follow just the emission of a phonon by an up electron. The absorption term is its Hermitian conjugate and the other spin is a second copy. This isolates the momentum transfer without expanding four terms related by symmetry:

In[1]:=
```
snegfermionoperators[c, ck]; snegbosonoperators[b, bk];
snegrealconstants[g]; sneglatticeindexes[{R}, Ns];
snegmomentumindexes[{k,q}, Ns];
emit = g sum[nc[number[c[R],UP], b[CR,R]], {R}];
emk = fourier[fourier[emit,c -> ck,R -> k], b -> bk,R -> q];
sumMomentumCanonical[sumExpand[emk]]
```

Out[6]=
$$\frac{g\sum_{k,q}\mathrm{bk}_q^\dagger \mathrm{ck}_{k\uparrow}^\dagger \mathrm{ck}_{k+q\uparrow}}{\sqrt{\mathrm{Ns}}}$$

The incoming electron momentum exceeds the outgoing one by q. The site sum has removed one independent momentum and left the local coupling divided by $\sqrt{N_s}$.

9.3 Displaced oscillator and induced attraction

For a fixed electronic occupation put $B = b + g(n - n_0)/\omega_0$. Then

$$\omega_0 b^\dagger b + g(n - n_0)(b + b^\dagger) = \omega_0 B^\dagger B - g^2(n - n_0)^2/\omega_0.$$

The ground energy depends on charge even when the bare phonon has no dispersion. The second difference between energies of charge 2, 1, 0 receives a shift $-2g^2/\omega_0$. In a hopping system a change of occupation also changes the displacement; the transformed hopping carries displacement operators. Replacing them by their phonon-vacuum expectations is an approximation. In an atomic charge sector the completion of the square is exact.

Case study: Convergence with the number of phonon quanta

Take $U = \omega_0 = 1$ and $g = 1/2$ in the Anderson–Holstein atom. In the empty sector the predicted lowest energy is $U/2 - g^2/\omega_0 = 1/4$. With a cutoff of two phonons inspect this sector alone. The off-diagonal elements should be $-g$ and $-\sqrt{2}g$:

In[1]:=
```
snegfermionoperators[d];
ham[n_] := isozsq[d[]]/2 + phononnumber[n] +
  nc[number[d[]]-1, phononx[n]]/2;
empty[n_] := First[keyselect[makematricesbzvc[ham[n],
  transformtoPH[qsbasisvc[{d[]},{n}]], {-1,0}]][[2]];
empty[2] // MatrixForm
```

Out[4]=
$$\begin{pmatrix} \frac{1}{2} & -\frac{1}{2} & 0 \\ -\frac{1}{2} & \frac{3}{2} & -\frac{1}{\sqrt{2}} \\ 0 & -\frac{1}{\sqrt{2}} & \frac{5}{2} \end{pmatrix}$$

Increasing the cutoff grows the variational space, so its lowest energy approaches the exact answer from above. High precision resolves a small truncation error that a rounded energy would hide:

In[5]:=
```
rows = Table[With[{e = Min[Eigenvalues[N[empty[n],35]]]},
  {n, N[e,8], N[-Log[10,e-1/4],5]}], {n,{1,2,4,8}}];
TableForm[rows, TableHeadings -> {None,{"Cutoff","Energy","-log10(
    error)"}}]
```

Out[6]=

Cutoff	Energy	-log10(error)
1	0.29289322	1.3676
2	0.25535571	2.2712
4	0.25002966	4.5278
8	0.25000000	10.146

The displaced vacuum contains a Poisson distribution in the original oscillator basis with mean $[g(n-n_0)/\omega_0]^2$. The largest occupied number in a truncated eigenvector is a useful convergence diagnostic in addition to its energy. On a dimer check the convergence of the bonding–antibonding splitting itself: it can be much smaller than the shift in an individual energy.

9.3.1 How a phonon cloud changes the hopping

Take one spinless electron and two local Einstein oscillators on a symmetric dimer. In the absence of tunnelling the two localized ground states are degenerate. Moving the electron interchanges which oscillator is displaced. Their phonon wave functions have overlap $e^{-(g/\omega_0)^2}$: each of two oscilla-

tors contributes $e^{-(g/\omega_0)^2/2}$. Degenerate perturbation theory gives a splitting $2te^{-(g/\omega_0)^2}$ to first order in t, with coupling $gn_j(b_j + b_j^\dagger)$ and real positive hopping. This splitting is a different observable from the single-site relaxation energy.

Build the matrix directly as a check on the phonon convention. The tensor order is electron site, phonon on the left, phonon on the right; t and g are in units of ω_0.

In[1]:=
```
split[n_,t_,g_] := Module[{id,nop,x,ph,h},
  id = IdentityMatrix[n+1]; nop = DiagonalMatrix[Range[0,n]];
  x = Normal[SparseArray[{Band[{1,2}] -> Sqrt[Range[n]],
                 Band[{2,1}] -> Sqrt[Range[n]]},{n+1,n+1}]];
  ph = KroneckerProduct[nop,id] + KroneckerProduct[id,nop];
  h = KroneckerProduct[IdentityMatrix[2],ph] -
      t KroneckerProduct[PauliX,id,id] +
      g (KroneckerProduct[DiagonalMatrix[{1,0}],x,id] +
         KroneckerProduct[DiagonalMatrix[{0,1}],id,x]);
  Differences[Take[Sort[Eigenvalues[N[h]]],2]][[1]]];
TableForm[Table[{n,split[n,0.01,0.6]/0.02, Exp[-0.6^2]}, {n,{2,4,6}}],

 TableHeadings -> {None,{"Cutoff","Splitting / 2t","Overlap"}}]
```

Out[2]=

Cutoff	Splitting / 2t	Overlap
2	0.726301	0.697676
4	0.698093	0.697676
6	0.697631	0.697676

Converging the cutoff at fixed t approaches the finite-hopping answer. Reducing t at fixed converged cutoff is the separate test of first-order perturbation theory.

9.4 Virtual phonons and coupled modes

At imaginary transferred frequency ν, eliminating a harmonic phonon coupled linearly to density yields $V(q, i\nu) = -2|g_q|^2\omega_q/(\nu^2 + \omega_q^2)$ for the dimensionless coordinate $b + b^\dagger$. Only for $|\nu| \ll \omega_q$ can this be replaced by a static attraction. A canonical transformation in powers of g must also avoid resonances between real electronic transitions and a phonon; there the eliminated

state belongs in the active space.

Case study: Stability of two coupled harmonic modes

$H = \omega(a^\dagger a + b^\dagger b) + \lambda(a^\dagger b^\dagger + ba)$ decomposes into the blocks $(a, b^\dagger)$ and $(b, a^\dagger)$ and is diagonalized symbolically; the energies are $\sqrt{\omega^2 - \lambda^2}$:

In[1]:=
```
snegbosonoperators[a, b];
snegrealconstants[w, lam];
{ε, T, Egs} = bogoliubov[w (number[a[]] + number[b[]]) + lam (nc[a[CR
    ], b[CR]] + nc[b[AN], a[AN]]), {a, b}];
{ε, Egs}
```

Out[4]= $\left\{\left\{\sqrt{w^2 - \text{lam}^2}, \sqrt{w^2 - \text{lam}^2}\right\}, \sqrt{w^2 - \text{lam}^2} - w\right\}$

For an orbital doublet coupled to two degenerate distortions the linear Jahn–Teller Hamiltonian at fixed coordinates contains $g(Q_1\tau_z + Q_2\tau_x)$. Its lower adiabatic surface is $K(Q_1^2 + Q_2^2)/2 - |g|\sqrt{Q_1^2 + Q_2^2}$. The continuum of classical minima does not imply a static symmetry-breaking expectation value in a finite quantum molecule: the kinetic energy of the nuclei mixes distortions. A finite two-mode Fock calculation can retain both.

Exercise 9.1 (*Relaxation energy*). Find the Jahn–Teller minimum radius and lowering of the energy. Relate the latter to the displaced-oscillator result for a single distortion quantized with $Q = (b + b^\dagger)/\sqrt{2M\omega}$.

10 Correlated electrons

What does it cost to move one electron between two hydrogen-like atoms? At large separation and with one electron on each atom, the final state contains an empty orbital and a doubly occupied one. Creating this pair costs the local repulsion U. Yet a virtual excursion into precisely the same state lowers the energy of a spin singlet. A charge excitation at energy U and a spin excitation at energy of order t^2/U thus have a common origin. The two-site Hubbard model lets us follow both without making an approximation. We will use its exact spectrum to judge two quite different approximations: eliminating charge fluctuations, and replacing the interaction by a self-consistent field.

The examples form one session. Start a fresh kernel with the SNEG paclet installed and evaluate the inputs in order. The convention for a level is `c[i]`; a single electron operator carries two additional indices, as in `c[CR,i,UP]`. Spin projections are in units of $\hbar$, and $k_{\mathrm{B}} = 1$.

10.1 One orbital: charge, spin and particle–hole symmetry

For one Wannier orbital on each of L sites we use

$$H = -t \sum_{\langle ij \rangle, \sigma} (c^{\dagger}_{i\sigma} c_{j\sigma} + c^{\dagger}_{j\sigma} c_{i\sigma}) + U \sum_{i} D_i, \qquad D_i = n_{i\uparrow} n_{i\downarrow}. \tag{10.1}$$

Here $n_{i\sigma} = c^{\dagger}_{i\sigma} c_{i\sigma}$, $\sigma = \uparrow, \downarrow$, and a bond occurs once in the sum. Throughout this chapter t is real and $U > 0$. In a crystal U is the interaction screened by degrees of freedom excluded from this orbital model; it need not equal an isolated-atom Coulomb integral. Our starting point is the atomic-limit construction in Ref. [3].

In[1]:=
```
Needs["sneg`"];
snegfermionoperators[c, d];
snegrealconstants[t, U, ε];
atom = ε number[d[]] + U hubbard[d[]];
{number[d[]], hubbard[d[]]}
```

Out[5]= $\{d_\downarrow^\dagger d_\downarrow + d_\uparrow^\dagger d_\uparrow, -d_\downarrow^\dagger d_\uparrow^\dagger d_\downarrow d_\uparrow\}$

The minus sign that may appear when D_i is printed in normal order is not an attractive interaction. Moving an annihilator through a creator of the other spin costs one interchange. A sign-independent way to read the result is to act on states. The following basis is labelled by $Q = N - 1$ and total spin S. Each sector of this atom contains a single multiplet; its energy is the sole entry of the block.

In[6]:=
```
ba = qsbasisvc[{d[]}];
rows = {Sequence @@ First[#], #[[2, 1, 1]]} & /@ makematricesbzvc[
    atom, ba];
TableForm[rows, TableHeadings -> {None, {"Q", "S", "Energy"}}]
```

Out[8]=

Q	S	Energy
-1	0	0
0	$\frac{1}{2}$	ϵ
1	0	$U + 2\epsilon$

The empty state has energy zero, the two spin states have energy ϵ, and $|2\rangle = d_\uparrow^\dagger d_\downarrow^\dagger |0\rangle$ has energy $2\epsilon + U$. `qsbasisvc` stores one representative of each spin multiplet: the dimension of a block counts copies of the representation, not its $2S + 1$ magnetic substates. With ϵ measured relative to the chemical potential, the atom has a spin-doublet ground state for $-U < \epsilon < 0$. At either endpoint two charge sectors cross; differentiating a single eigenvalue through such a crossing would give the wrong ground-state occupation.

At half filling a particle–hole transformation on a bipartite lattice takes $c_{i\sigma} \mapsto \eta_i c_{i\sigma}^\dagger$, with $\eta_i = +1$ on one sublattice and -1 on the other. Hopping is invariant because neighbours have opposite η and distinct fermions anticommute. Locally $n_i \mapsto 2 - n_i$ and $D_i \mapsto 1 - n_i + D_i$. Thus the grand Hamiltonian with $\epsilon = -U/2$ is invariant. Equivalently,

$$U(n_{i\uparrow} - \tfrac{1}{2})(n_{i\downarrow} - \tfrac{1}{2}) = UD_i - \tfrac{U}{2} n_i + \tfrac{U}{4}.$$

Dropping the last term changes the energy origin; dropping the term linear in n_i also changes the chemical potential. Neither affects splittings at fixed N, but only a constant can be ignored when comparing absolute addition and removal energies at a specified μ.

10.1.1 A free spin without a static polarization

At $\epsilon = -U/2$ the empty and doubly occupied states both have energy zero and the two singly occupied states each have energy $-U/2$. For $T > 0$ the grand partition function is $2(1 + e^{U/(2T)})$. It follows that the mean occupation is exactly one at *any* temperature; it tells us nothing about moment formation. A more discriminating observable is $4\langle(S^z)^2\rangle$, which equals the probability of single occupation. The complementary observable, $\langle(n-1)^2\rangle$, counts empty and doubly occupied sites. Test the two in a thermal trace over all four charge and spin states:

In[9]:=
```
sz2 = pow[number[d[], UP] - number[d[], DO], 2];
q2 = pow[number[d[]] - 1, 2];
Expand[sz2 + q2]
```

Out[11]= 1

In[12]:=
```
hat = atom /. {ε -> -1, U -> 2};
rows = Table[Prepend[Round[thermalaverage[#, hat, {d}, temp], 1/1000]
     & /@
    {number[d[]], sz2, q2}, temp], {temp, {0, 1, 4}}];
TableForm[N[rows], TableHeadings -> {None, {"T", "Charge", "Moment",
    "Charge variance"}}]
```

Out[14]=

T	Charge	Moment	Charge variance
0.	1.	1.	0.
1.	1.	0.731	0.269
4.	1.	0.562	0.438

The columns are T, $\langle n\rangle$, $4\langle(S^z)^2\rangle$ and $\langle(n-1)^2\rangle$, for $U = 2$ in the energy units used here. At zero temperature `thermalaverage` averages equally over the two ground states, so $\langle S^z\rangle = 0$ although the spin moment is maximal. A field term $-hS^z$ lifts that degeneracy. Since S^z commutes with the atomic Hamiltonian, the static susceptibility at $h = 0$ is $\chi = \langle(S^z)^2\rangle/T$ (the magnetic

moment multiplying the field is set to one). At low nonzero T the free-spin contribution $1/(4T)$ survives; at high T the two spinless states halve its coefficient. Taking $T \to 0$ at fixed $h > 0$ selects a pure spin state with a different magnetization from the zero-field mixture.

10.2 Exact solution of the Hubbard dimer

Connect two sites by *one* bond. The empty and doubly occupied local states have spin zero, so each can mix with a two-site singlet but not with a triplet. Build the full Hamiltonian before restricting the basis. Products of operators are made with `nc`; ordinary multiplication suffices for a scalar times an operator. This construction recasts the two-site symmetry problem of Steeb and Hardy [4, Sec. 3.4, problem 84] in a basis of total spin instead of fixed spin projection.

In[15]:=
```
sites = {c[1], c[2]};
h0 = U Total[hubbard[sites]];
v = -t hop[c[1], c[2]];
hd = h0 + v;
Expand[{komutator[hd, Total[number /@ sites]],
  komutator[hd, Total[spinplus /@ sites]]}]
```

Out[19]= $\{0, 0\}$

The second zero checks more than conservation of S^z: together with Hermiticity it establishes invariance under spin rotations. At $L = 2$, $Q = N - 2$, so the half-filled blocks are those with $Q = 0$.

In[20]:=
```
bd = qsbasisvc[sites];
sector[s_] := Select[bd, First[#] == {0, s} &];
mat[h_, b_] := First[makematricesbzvc[h, b]][[2]];
hs = mat[hd, sector[0]];
{Dimensions[hs], mat[hd, sector[1]]}
```

Out[24]= $\{\{3, 3\}, \begin{pmatrix} 0 \end{pmatrix}\}$

In[25]:=
```
hs // MatrixForm
```

Out[25]=
$$\begin{pmatrix} U & -\sqrt{2}t & 0 \\ -\sqrt{2}t & 0 & -\sqrt{2}t \\ 0 & -\sqrt{2}t & U \end{pmatrix}$$

In this phase convention the diagonal states, in order, are a doublon on one site, the spin singlet with one electron on each site, and a doublon on the other site. Each amplitude $-\sqrt{2}\,t$ is the sum of two indistinguishable hopping paths out of the normalized singlet. Rephasing one basis vector flips a row and the corresponding column; it cannot change an eigenvalue. With $P(x) = \det(H_{S=0} - xI)$:

In[26]:=
```
Factor[CharacteristicPolynomial[hs, x]]
```

Out[26]=
$$-\left((U - x)\left(4t^2 + Ux - x^2\right)\right)$$

The antisymmetric doublon combination decouples at energy U. The symmetric combination couples to the singly occupied singlet by $-2t$. The other two singlet energies and the singlet–triplet splitting are

$$E_{\pm} = \frac{U \pm \sqrt{U^2 + 16t^2}}{2}, \qquad \Delta_s = E_T - E_- = -E_-. \tag{10.2}$$

At $U = 0$ the singlet fills the bonding orbital with both spins and has energy $-2|t|$. At $t = 0$ the singly occupied singlet and the triplet are degenerate. Turning on hopping at large U leaves the triplet at zero but lowers the singlet. This is the simplest example of kinetic energy selecting antiferromagnetic correlations.

10.2.1 What remains of double occupation?

The derivative $\partial E_-/\partial U = \langle D_1 + D_2 \rangle$ counts the doublons in the exact ground state. The equality follows by differentiating the eigenvalue equation for a normalized eigenvector; terms involving its derivative cancel. It is valid away from a crossing.

In[27]:=
```
emin = (U - Sqrt[U^2 + 16 t^2])/2;
dsite = Simplify[D[emin, U]/2];
{dsite, dsite /. {U -> 0, t -> 1}}
```

Out[29]= $\{\frac{1}{4}\left(1-\frac{U}{\sqrt{16t^2+U^2}}\right),\frac{1}{4}\}$

For $U \gg |t|$ the probability per site is $2t^2/U^2$ to leading order. It is small but cannot be set to zero before evaluating the energy. In fact $\langle v\rangle = t\,\partial E_-/\partial t$, and at this order the kinetic gain is twice the interaction cost. Setting $U = 4t$ gives an instructive intermediate case: the exact singlet still has no local magnetization, even though its local moment $\langle(n_{i\uparrow}-n_{i\downarrow})^2\rangle = 1-2\langle D_i\rangle$ already substantially exceeds the free-electron value $1/2$.

Pitfall

Occupation vectors refer to the current list of modes. A new call to a basis generator can change it. Construct the matrix before creating a basis on different sites, or regenerate the original basis first. The numerical matrix, once constructed, is independent of that list.

10.3 Eliminating virtual charge excitations

Let P select one electron per site and $\bar{P} = 1 - P$. In the half-filled dimer $PH_0P = PvP = 0$. Every intermediate state connected to P by one hop has energy U, so to second order

$$H^{(2)} = -Pv\bar{P}vP/U.$$

It is negative semidefinite. In a spin triplet there is no accessible doublon, hence no shift. For a singlet the squared norm of $\bar{P}v|s\rangle$ is $4t^2$. Since the singlet projector is $1/4 - \mathbf{S}_1\cdot\mathbf{S}_2$, this already fixes the entire spin Hamiltonian.

In[30]:=
```
p = nc @@ (projector1 /@ sites);
heff = schriefferwolff[h0, v, p, 2];
snegxoperators[X, "HubbardAtom"];
snegspinoperators[{S, 1/2}];
Expand[xtospin[fermionstox[heff[[3]], c -> X], X -> S]]
```

Out[34]= $-\frac{t^2}{U}+\frac{4S_1^zS_2^zt^2}{U}+\frac{2S_1^+S_2^-t^2}{U}+\frac{2S_1^-S_2^+t^2}{U}$

Read the transverse terms with $S_1^xS_2^x+S_1^yS_2^y = (S_1^+S_2^- + S_1^-S_2^+)/2$. The result

is $J(\mathbf{S}_1 \cdot \mathbf{S}_2 - 1/4)$ with $J = 4t^2/U$. `heff[[3]]` is order two: the returned list starts with order zero. The constant $-J/4$ matters when comparing energies rather than splittings. A controlled expansion requires $|t|/U \ll 1$. The exact dimer shows both the sign and scale of the first omitted term:

In[35]:=
```
Normal[Series[-emin, {t, 0, 4}, Assumptions -> U > 0]] //
  Simplify[#, U > 0] &
```

Out[35]=
$$\frac{4t^2\left(U^2 - 4t^2\right)}{U^3}$$

Figure 10.1 compares the two truncations. The fourth-order correction lowers the second-order splitting. At $U = 2t$ it cancels the entire leading term, although the exact gap is positive. The relevant scale is visible already in Eq. (10.2): the Taylor series of the square root about $t = 0$ converges for $|4t/U| < 1$, not throughout the range in the figure.

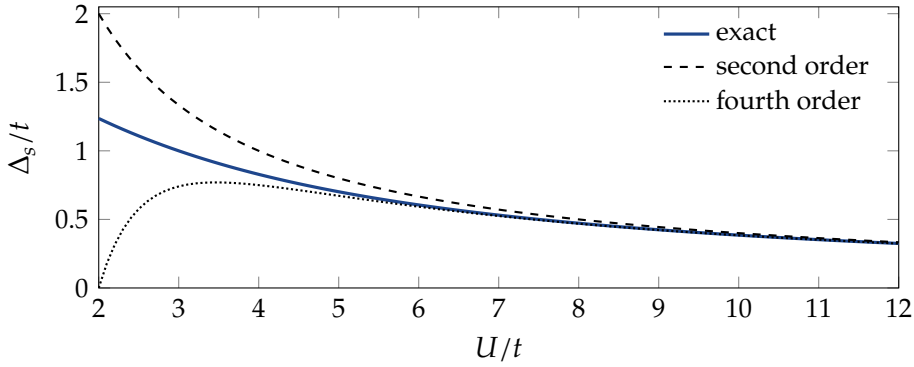


Figure 10.1: Singlet–triplet gap of a half-filled dimer, $t > 0$. All curves use the same one-bond hopping. At large U/t the remaining error after the fourth-order correction is of order $(t/U)^5$ in the plotted units.

10.3.1 Doping: when the first-order term survives

Remove an electron. The low-energy subspace must now include holes, so use $P_G = \prod_i (1 - D_i)$ rather than the projector onto exactly one electron per site.

A hop onto an empty neighbour need not leave P_G. With $\tilde{c}_{i\sigma} = c_{i\sigma}(1 - n_{i\bar{\sigma}})$, the usual nearest-neighbour t–J Hamiltonian is

$$H_{tJ} = -t \sum_{\langle ij\rangle\sigma} (\tilde{c}^{\dagger}_{i\sigma}\tilde{c}_{j\sigma} + \text{h.c.}) + J \sum_{\langle ij\rangle} (\mathbf{S}_i \cdot \mathbf{S}_j - \tfrac{1}{4} n_i n_j). \tag{10.3}$$

The density factor makes the exchange vanish if either site is empty. On three or more sites this expression does *not* exhaust order t^2/U. A hop $k \to j$ can create a doublon, followed by a hop $j \to i$ that removes it with $i \neq k$. If i was initially empty, the final state is in P_G, but the hole has moved from i to k. For real nearest-neighbour hopping the extra term can be written

$$-\frac{t^2}{U} \sum_{j,\, i \neq k \in \mathcal{N}(j),\, \sigma} P_G (c^{\dagger}_{i\sigma} n_{j\bar{\sigma}} c_{k\sigma} - c^{\dagger}_{i\sigma} c^{\dagger}_{j\bar{\sigma}} c_{j\sigma} c_{k\bar{\sigma}}) P_G .$$

The neighbours (i, k) are ordered; the reverse path is already in the sum. The minus sign inside the parentheses distinguishes the two possible spin transfers. At half filling both transfers to a third site are blocked by the final projector. Away from half filling, neglecting them is an additional modelling decision, not a higher-order truncation.

Isolating the three-site matrix element. Take an open chain 1–2–3 with two electrons in a spin singlet. There are three states with the electrons on different sites and three with both electrons on one site. A singlet shared by sites 1 and 2 can turn into a singlet on 2 and 3 in two hops through a doublon on site 2. In contrast, exchange on either individual bond cannot change the hole position. This makes the off-diagonal element of $H^{(2)}$ between the two bond singlets a diagnostic for the three-site term.

In[36]:=
```
line3 = c /@ Range[3];
b3 = Select[qsbasisvc[line3], First[#] == {-1, 0} &];
nd = Diagonal[mat[Total[hubbard[line3]], b3]];
lo = Flatten[Position[nd, 0]]; hi = Flatten[Position[nd, 1]];
v3 = mat[-hop[c[1], c[2]] - hop[c[2], c[3]], b3];
k3 = -v3[[lo, hi]] . v3[[hi, lo]];
k3 // MatrixForm
```

Out[42]=
$$\begin{pmatrix} -4 & 0 & -2 \\ 0 & 0 & 0 \\ -2 & 0 & -4 \end{pmatrix}$$

The printed matrix is in units of t^2/U; the dimensionless entries are exact. A pair occupying sites 1 and 3 has no virtual doublon one hop away, hence a zero row. The two adjacent singlets have diagonal shift -4 and mutual coupling of magnitude 2: only the intermediate doublon on the shared site connects them. Which row is zero and the sign of the coupling depend on the chosen basis. The eigenvalues do not. Removing the three-site term from this matrix gives a spurious degeneracy:

In[43]:=
```
{Sort[Eigenvalues[k3]], Sort[Diagonal[k3]]}
```

Out[43]=
$$\begin{pmatrix} -6 & -2 & 0 \\ -4 & -4 & 0 \end{pmatrix}$$

These numbers diagnose the second-order *operator*, not the full energy shifts of the doped chain. To obtain the low-energy spectrum through this order, add the first-order block $t\, v_{3,\mathrm{lo},\mathrm{lo}}$ to $(t^2/U)\, k_3$ and diagonalize. At half filling the first-order block would be zero.

10.3.2 Four sites: a process with no dimer counterpart

Four successive hops around a square can restore single occupation while permuting all four spins. On a bipartite plaquette, returning to single occupation requires an even number of hops, so the first corrections to superexchange occur at order t^4/U^3. They include both pair exchanges and a cyclic permutation plus its inverse [5]. Renormalizing J alone cannot describe an entire plaquette spectrum. A coefficient quoted for a four-spin product is not directly the coefficient of a permutation: expanding a spin-1/2 swap, $P_{ij} = 2\mathbf{S}_i \cdot \mathbf{S}_j + 1/2$, generates lower-body terms too.

The plaquette example compares the lowest sixteen half-filled energies with the sixteen energies in the projected spin space. From the checkout run:
`wolframscript -file examples/schriefferwolff-plaquette.m`
Keeping t fixed and doubling large U reduces the leading absolute spectral error by about eight at second order (t^4/U^3) and by about thirty-two at fourth order (t^6/U^5). Matching the dimension and the energy origin is essential in

this comparison.

Further reading

For the passage from electrons to spins see Auerbach [6]. In a junction the virtual states can include superconducting quasiparticles; their energies replace the single denominator U of the dimer. An application is the Josephson harmonic expansion of Bácsi, Iličin and Žitko [7].

10.4 Measuring a charge gap on a finite cluster

Define $E_0(N)$ as the lowest energy among *all* spin sectors at particle number N. At zero temperature that charge sector is stable over a chemical-potential interval of width

$$\Delta_c = \mu_+ - \mu_- = E_0(N+1) + E_0(N-1) - 2E_0(N). \tag{10.4}$$

This second difference eliminates every term linear in N. It is not the energy of the lowest neutral excitation: the dimer, for instance, admits an arbitrarily soft spin excitation as $t^2/U \to 0$, even as its charge gap grows.

Take four sites on a ring, count each bond once, and measure energies in units of $t > 0$. Here we keep S^z, rather than total S, and use all magnetic substates. The blocks are larger but their dimensions now sum directly to the Fock-space dimension 4^4.

In[44]:=
```
ring = c /@ Range[4];
hr = -Sum[hop[c[i], c[Mod[i, 4] + 1]], {i, 4}] +
  U Total[hubbard[ring]];
br = qszbasisvc[ring];
blocks = makematricesbzvc[hr, br];
{Length[blocks], Total[Length /@ blocks[[All, 2]]]}
```

Out[48]= {25, 256}

In[49]:=
```
e0[q_, u_] := Min[Flatten[Eigenvalues[N[#[[2]] /. U -> u]] & /@
  Select[blocks, #[[1, 1]] == q &]]];
gap[u_] := e0[1, u] + e0[-1, u] - 2 e0[0, u];
rows = Table[{u, N[Round[gap[u], 1/1000]]}, {u, {0, 2, 4, 8, 16}}];
TableForm[rows, TableHeadings -> {None, {"U/t", "Charge gap / t"}}]
```

Out[52]=

U/t	Charge gap / t
0	0.
2	1.238
4	2.701
8	5.991
16	13.361

Each row gives $(U/t, \Delta_c/t)$ rounded to three decimal places. At $U = 0$ the allowed momenta are $0, \pm\pi/2, \pi$ (lattice spacing one). The two zero-energy orbitals make this an open shell, so adding and removing an electron have identical one-particle thresholds. The same calculation on a closed-shell cluster can yield a nonzero gap in a metal. Inferring a bulk gap requires a sequence of sizes and control of such shell effects.

For comparison, at $U = 0$ alternate the hopping amplitudes $t_1, t_2 > 0$ in a chain with two sites per cell. The Bloch energies are $\pm|t_1 + t_2 e^{ik}|$, where k is in inverse cell lengths. The half-filled band gap $2|t_1 - t_2|$ persists in the thermodynamic limit, without an interaction. In the large-U uniform Hubbard chain charge motion instead costs a doublon energy even in the absence of a one-particle band gap. The charge-gap measurement alone cannot tell these mechanisms apart; compare the noninteracting spectrum, double occupancy and spin response.

10.4.1 A shell-effect check without many-body diagonalization

The noninteracting uniform ring is a useful calibration. Sort its L one-particle levels *per spin*; at half filling each spin occupies $L/2$ levels. The finite-size addition gap is the difference of two consecutive levels at the Fermi surface.

In[53]:=

```
shell[l_] := With[{band = SortBy[Table[-2 Cos[2 Pi j/l], {j, 0, l-1}],
     N]},
  band[[l/2 + 1]] - band[[l/2]]];
TableForm[Table[{l, N[shell[l]]}, {l, {4, 6, 8, 10}}],
  TableHeadings -> {None, {"Sites", "Charge gap / t"}}]
```

Out[54]=

Sites	Charge gap / t
4	0.
6	2.
8	0.
10	1.23607

For $L = 4p$, both Fermi points lie on the grid and the gap is zero. For $L = 4p + 2$, the grid misses each Fermi point by π/L, so $\Delta_c/t = 4\sin(\pi/L)$: a finite gap at *every* size on this subsequence tends to zero as $1/L$. In an interacting calculation one cannot simply subtract this formula; the velocities and the many-body levels have changed. But treating the two subsequences separately prevents an alternation of boundary-condition effects from masquerading as an interaction threshold.

10.5 The atomic propagator and Hubbard-I

Consider the retarded one-electron Green's function $G_\sigma(z) = \langle\langle d_\sigma; d_\sigma^\dagger \rangle\rangle_z$, where $z = \omega + i0^+$. Its spectral density $A_\sigma(\omega) = -\operatorname{Im} G_\sigma(\omega + i0^+)/\pi$ has unit integral *per spin*. Use the same grand Hamiltonian as in the atomic spectrum. Since $n_{\bar\sigma}$ commutes with it, the two operators $d_\sigma(1 - n_{\bar\sigma})$ and $d_\sigma n_{\bar\sigma}$ evolve at fixed frequencies ϵ and $\epsilon + U$, respectively. The equation-of-motion hierarchy closes:

$$(z - \epsilon) G_\sigma = 1 + U F_\sigma,$$
$$(z - \epsilon - U) F_\sigma = \langle n_{\bar\sigma} \rangle,$$

where $F_\sigma = \langle\langle d_\sigma n_{\bar\sigma}; d_\sigma^\dagger \rangle\rangle_z$. These are operator identities; no factorization of an average was used.

In[55]:=
```
gatom = greensfunction[d[AN, UP], d[CR, UP], atom, z];
g = gatom /. avg[number[d[], DO]] -> w;
Apart[g, z]
```

Out[57]=

$$\frac{1 - w}{z - \epsilon} + \frac{w}{-U + z - \epsilon}$$

Writing $w = \langle n_\downarrow \rangle$, the poles and their weights are transparent in

$$G_\uparrow(z) = \frac{1 - w}{z - \epsilon} + \frac{w}{z - \epsilon - U}. \tag{10.5}$$

The upper pole exists only when a down electron is available on the site. With $\epsilon = -U/2$ and a spin-symmetric ensemble, $w = 1/2$; the spectral peaks then have equal weight and are separated by U. A pure member of the atomic doublet has the same energy but different spin-resolved weights. Substituting $w = 1/2$ is a specification of the state, not a step in the equation-of-motion calculation.

In[58]:=
```
ph = Factor[g /. {ε -> -U/2, w -> 1/2}];
{ph, Limit[z ph, z -> Infinity]}
```

Out[59]= $\{\dfrac{4z}{(2z-U)(U+2z)}, 1\}$

Checking both residues against a trace. The half-filled atom is a poor test of thermal weights: symmetry forces them both to $1/2$ regardless of temperature. Instead take $\epsilon = -1/4$, $U = 2$, and $T = 1/2$. Removing or adding one spin-up electron still gives only the two transition frequencies ϵ and $\epsilon + U$, but their residues differ. The Lehmann representation sums matrix elements between exact many-body eigenstates. Each row below is $(E_n - E_m, w_{mn})$ after collecting equal frequencies:

In[60]:=
```
asym = atom /. {ε -> -1/4, U -> 2};
poles = lehmann[d[AN, UP], d[CR, UP], asym, {d}, 1/2];
TableForm[N[Round[poles, 1/1000]], TableHeadings -> {None, {"
    Frequency", "Residue"}}]
```

Out[62]=

Frequency	Residue
−0.25	0.609
1.75	0.391

For a retarded anticommutator the factor in the residue is the *sum* of the two Boltzmann probabilities, not their difference. The atomic partition function is $Z = 1 + 2e^{-\epsilon/T} + e^{-(2\epsilon+U)/T}$. The upper peak has weight $w = (e^{-\epsilon/T} + e^{-(2\epsilon+U)/T})/Z$. These facts give two independent checks: the total weight is one, and the upper weight equals the mean down-spin occupation. Compare the full complex propagators at a frequency off the real axis, where a pole neither diverges nor needs artificial broadening:

In[63]:=
```
wth = thermalaverage[number[d[], DO], asym, {d}, 1/2];
z0 = 1/3 + I/4;
gtrace = Total[poles[[All, 2]]/(z0 - poles[[All, 1]])];
{Abs[1 - Total[poles[[All, 2]]]] < 10^-12,
 Abs[gtrace - (g /. {z -> z0, ε -> -1/4, U -> 2, w -> wth})] <
    10^-12}
```

Out[66]= {True, True}

10.5.1 What changes when the atom is put back in the lattice?

Define the local self-energy by $G_{\mathrm{at}}^{-1}(z) = z - \epsilon - \Sigma_{\mathrm{at}}(z)$. Inverting Eq. (10.5) gives

$$\Sigma_{\mathrm{at}}(z) = Uw + \frac{U^2 w(1-w)}{z - \epsilon - U(1-w)}.$$

The static term is the Hartree shift. The pole in the second term is what opens a correlation gap in this approximation. Hubbard-I inserts this atomic self-energy into the lattice propagator, or equivalently uses $G_k(z) = [G_{\mathrm{at}}^{-1}(z) - \varepsilon_k]^{-1}$, with ε_k the dispersion of the hopping term alone.

In[67]:=
```
gk = Together[ph/(1 - ek ph)];
Factor[Denominator[gk]]
```

Out[68]= $-U^2 - 4\mathrm{ek}z + 4z^2$

At particle–hole symmetry its poles are $z_\pm = (\varepsilon_k \pm \sqrt{\varepsilon_k^2 + U^2})/2$. For $U > 0$ they lie on opposite sides of zero. For $U = 0$ the apparent additional pole cancels against a zero in the numerator, leaving $1/(z - \varepsilon_k)$. Checking residues as well as pole locations prevents a spurious flat band. The construction is exact in the atomic limit and recovers free propagation at $U = 0$. It omits intersite spin correlations and has no coherent Fermi-liquid resonance. Its half-filled gap at arbitrarily small $U > 0$ does not establish a Mott transition for a general lattice.

10.6 Can a static field account for the singlet energy?

In a collinear Hartree state, write $n_{i\sigma} = \bar{n}_{i\sigma} + \delta n_{i\sigma}$ and discard $\delta n_{i\uparrow}\delta n_{i\downarrow}$. The remaining interaction is

$$U \sum_i (\bar{n}_{i\downarrow} n_{i\uparrow} + \bar{n}_{i\uparrow} n_{i\downarrow} - \bar{n}_{i\uparrow}\bar{n}_{i\downarrow}). \tag{10.6}$$

The last term does not change the orbitals, but omitting it counts the interaction energy twice. In a translationally invariant metal, a small uniform polarization $m = \bar{n}_\uparrow - \bar{n}_\downarrow$ splits the two spin bands by Um. For a smooth one-spin density of states $\rho(E_F)$ per site, the induced polarization at zero temperature is $\rho(E_F)Um$ to first order. Thus the unpolarized solution is unstable to *uniform* magnetization for $U\rho(E_F) > 1$ (Stoner criterion). It need not be the first instability: the susceptibility at nonzero wave vector can be larger.

On the dimer, try staggered polarization instead: $m_1 = -m_2 = m$, with one electron per site on average. Apart from the common Hartree shift, each spin sees diagonal potentials $\pm Um/2$ and off-diagonal hopping $-t$. Filling the lower eigenstate for each spin gives

$$m = \frac{Um/2}{\sqrt{t^2 + (Um/2)^2}}.$$

Besides $m = 0$, for $U > 2|t|$ there are two solutions related by spin reversal with $m = \pm\sqrt{1 - (2t/U)^2}$. Below we seed one branch; the solver adjusts the chemical potential for two particles.

In[69]:=
```
hf = meanfield[hd /. {t -> 1, U -> 4}, "Hartree"];
seed = {avg[number[c[1], UP]] -> 0.9, avg[number[c[2], DO]] -> 0.9};
sol = meanfieldsolve[hf, sites, seed, "ParticleNumber" -> 2];
m1 = avg[number[c[1], UP]] - avg[number[c[1], DO]];
{sol["Converged"], Round[sol["Energy"], 1/1000],
  Round[m1 /. sol["Averages"], 1/1000] // N}
```

Out[73]= $\{\text{True}, -\frac{1}{2}, 0.866\}$

At these parameters the unpolarized determinant has energy $-2t + U/2 = 0$, the staggered determinant has energy $-t/2$, and the exact energy is $(2 - 2\sqrt{2})t$.

Improving the variational energy by allowing a staggered field has introduced a false broken symmetry on a finite dimer. In particular m_1 is not the local fluctuating moment calculated in the singlet. At large U, Hartree theory yields $-2t^2/U$ for this determinant; the exact singlet gains twice as much through transverse spin exchange. Convergence of the iteration is not a test of either the symmetry or the accuracy of the state.

10.7 Local constraints: Hubbard operators and auxiliary particles

Operator projection is easiest to understand on one site. For $a, b \in \{0, \uparrow, \downarrow, 2\}$ let $X^{ab} = |a\rangle\langle b|$. Then $X^{ab}X^{cd} = \delta_{bc}X^{ad}$ and $\sum_a X^{aa} = 1$. The creation operator $d^\dagger_\uparrow$ is the sum of the transitions $X^{\uparrow 0} + X^{2\downarrow}$. For down spin the second transition has a minus sign because $|2\rangle$ was defined with the up creator on the left. Work with the corresponding fermionic polynomials to test the projected algebra:

In[74]:=
```
xi = xoperator[d[], "0", "UP"];
Expand[antikomutator[xi, conj[xi]]]
```

Out[75]= $1 - d^\dagger_\downarrow d_\downarrow$

The anticommutator is $X^{00} + X^{\uparrow\uparrow} = 1 - n_\downarrow$, not the identity. The missing weight is precisely the transition at the upper Hubbard pole. At distinct sites, odd transitions anticommute: ordinary commuting site matrices would lose the fermion sign.

In[76]:=
```
pg = 1 - hubbard[d[]];
Expand[{nc[pg, pg] - pg, nc[pg, d[AN, UP], pg] - xi}]
```

Out[77]= $\{0, 0\}$

One can instead keep canonical auxiliary fields at the price of a constraint. In the no-doublon space introduce a boson b for $|0\rangle$ and fermions f_σ for $|\sigma\rangle$. Identify $X^{0\sigma} = b^\dagger f_\sigma$ and impose

$$b^\dagger b + \sum_\sigma f^\dagger_\sigma f_\sigma = 1 \quad \text{on every site.}$$

The three physical states are precisely the states with one auxiliary particle.

The substitutions $b \to e^{i\theta} b$ and $f_\sigma \to e^{i\theta} f_\sigma$ leave $X^{0\sigma}$ invariant: the common phase is redundant. Discarding the constraint introduces, among others, an auxiliary vacuum and a two-spinon state with no counterparts in the projected atom. Enforcing it only on average is a mean-field approximation. The idempotent P_G and the Hubbard operators above implement the projection exactly and offer local tests for any auxiliary-particle calculation.

10.7.1 Partial projection as a variational wave function

What is lost if one discards doublons altogether? Return to the dimer, with $t > 0$ and exactly two particles. The free bonding determinant is an equal-amplitude superposition of the singly occupied singlet $|s\rangle$ and the symmetric doublon $|D_+\rangle$, in phases chosen so that $\langle s|H|D_+\rangle = -2t$. Multiply each configuration with a doublon by g, $0 \le g \le 1$. This is a small-system version of a Gutzwiller variational state:

$$|\psi(g)\rangle = \frac{|s\rangle + g|D_+\rangle}{\sqrt{1+g^2}}, \qquad E(g) = \frac{-4tg + Ug^2}{1+g^2}.$$

There is no approximation in evaluating this expectation value. More unusually, on the dimer this one-parameter family contains the exact ground state. Stationarity gives $2tg^2 + Ug - 2t = 0$, whose positive root can be written without subtracting nearly equal numbers:

In[78]:=
```
gopt = 4 t/(U + Sqrt[U^2 + 16 t^2]);
energy[y_] := (-4 t y + U y^2)/(1 + y^2);
FullSimplify[{energy[gopt] - emin,
  gopt^2/(2 (1 + gopt^2)) - dsite}, U > 0 && t > 0]
```

Out[80]= $\{0, 0\}$

The second check is the probability of finding a doublon on one specified site. At $t/U \ll 1$ the optimum has $g \simeq 2t/U$, whereas strict projection $g = 0$ gives $E = 0$ when evaluated with the original Hamiltonian. A nonzero admixture of *amplitude* t/U produces exchange energy of order t^2/U. The projected spin Hamiltonian recovers that energy by transforming the Hamiltonian, rather than by retaining the doublon in the state. For larger clusters inequivalent hole–doublon separations generally require different amplitudes; one

scalar g is no longer enough to make the trial state exact.

10.8 A three-orbital atom: Hund's rule from the spectrum

Distinguish a single-particle level index from a site index: now $a_{m\sigma}$ removes an electron in orbital $m = 1, 2, 3$ of *one* atom. Set all three orbital energies to zero. For a rotationally invariant Kanamori interaction the repulsion in distinct orbitals is $U' = U - 2J$ for opposite spin and $U' - J$ for equal spin. Density terms alone would give different energies to the three components of a spin triplet. Two additional processes restore the multiplet structure:

$$H_K = U \sum_m D_m + (U - 2J) \sum_{m \neq n} n_{m\uparrow} n_{n\downarrow} + (U - 3J) \sum_{m<n,\sigma} n_{m\sigma} n_{n\sigma} - J \sum_{m \neq n} S_m^+ S_n^- + J \sum_{m \neq n} B_m^\dagger B_n, \qquad B_m = a_{m\downarrow} a_{m\uparrow}. \tag{10.7}$$

The ordered last sums already contain their Hermitian conjugates. The pair convention matters: reversing only the two annihilators would reverse the pair-hopping amplitude. To make all factors explicit, construct the terms for each unordered orbital pair:

In[81]:=
```
snegfermionoperators[a];
snegrealconstants[J];
orb = a /@ Range[3];
n[m_, s_] := number[a[m], s];
pair[m_] := nc[a[AN, m, DO], a[AN, m, UP]];
hkpair[m_, k_] := (U - 2 J) (nc[n[m, UP], n[k, DO]] +
    nc[n[m, DO], n[k, UP]]) +
  (U - 3 J) Total[nc[n[m, #], n[k, #]] & /@ {UP, DO}] -
  J (nc[spinplus[a[m]], spinminus[a[k]]] +
     nc[spinminus[a[m]], spinplus[a[k]]]) +
  J (nc[conj[pair[m]], pair[k]] + nc[conj[pair[k]], pair[m]]);
hk = U Total[hubbard[orb]] + Total[hkpair @@@ Subsets[Range[3],
    {2}]];
Expand[{hk - conj[hk], komutator[hk, Total[spinplus /@ orb]]}]
```

Out[88]= $\{0, 0\}$

Put two electrons in six spin-orbitals. Here $Q = N - 3 = -1$. The following rows give S, the energy (in symbolic U, J), and the number of *multiplets* at

that energy. We keep the multiplicities; distinct eigenvalues alone cannot diagnose a missing state.

In[89]:=
```
bk = qsbasisvc[orb];
k2 = makematricesbzvc[hk, Select[bk, #[[1, 1]] == -1 &]];
levels = Flatten[Table[{b[[1, 2]], #[[1]], #[[2]]} & /@
    Tally[Expand /@ Eigenvalues[b[[2]]]], {b, k2}], 1];
TableForm[Sort[levels], TableHeadings -> {None, {"Spin", "Energy", "
    Multiplets"}}]
```

Out[92]=

Spin	Energy	Multiplets
0	$U - J$	5
0	$2J + U$	1
1	$U - 3J$	3

For $J > 0$ the lowest energy is $U - 3J$. The two electrons occupy different orbitals with parallel spins; three choices of orbital pair give three triplets. Among singlets, each distinct-orbital pair gives one state at $U - J$. The three intra-orbital pairs form a matrix with diagonal U and off-diagonal J. Its symmetric vector has energy $U + 2J$, and the two orthogonal vectors have energy $U - J$. Thus the dimensions are $9 + 5 + 1 = \binom{6}{2}$. Dropping pair hopping would leave three degenerate states at U instead of this $1 + 2$ splitting.

In[93]:=
```
Total[(2 #[[1]] + 1) #[[3]] & /@ levels] == Binomial[6, 2]
```

Out[93]= True

Further reading

The atomic splittings determine which spin and orbital degrees of freedom survive when the shell couples to a bath. For the three-orbital impurity with this interaction see Horvat, Žitko and Mravlje [8]. The authors' publication list provides access to the paper.

Exercises

Exercise 10.1 (*Counting charge defects*). At $t = 0$ count the half-filled ground states and the states exactly U higher on $L \geq 2$ sites. How many states must a

charge-only basis contain at each energy for $L = 4$?

Exercise 10.2 (*Detuned sites*). Add $\delta(n_2 - n_1)/2$ to the half-filled dimer. Derive the exchange $J(\delta)$ to second order in t and state the condition on U, δ, t for this spin-only description. Compare with a singlet eigenvalue, taking $U = 10$, $\delta = 6$ and $t = 1/10$ in common energy units.

Exercise 10.3 (*Addition versus spin gap*). At half filling find the dimer charge gap for all $U \geq 0$, $t > 0$. Why does it remain positive at $U = 0$? Compare it with the spin gap at $U = 0$ and at $U/t \gg 1$.

Exercise 10.4 (*A spectral sum rule*). Derive the first three high-frequency coefficients of the atomic propagator at general filling. Which of them distinguish the two Hubbard poles from a single Hartree pole? Verify with nested commutators without extracting the poles a second time.

11 Superconductivity

11.1 From a Cooper pair to the gap equation

Above an inert Fermi sea two electrons with opposite momenta can scatter repeatedly within a shell of width ω_D. For a constant attraction $-g$ (with the volume absorbed in the normalization of the sum), their binding energy $E_b > 0$ satisfies $1 = g \sum_{0<\xi_k<\omega_D} (2\xi_k + E_b)^{-1}$. A nonzero one-spin density of states at the Fermi surface makes this sum logarithmic as $E_b \to 0$. The instability is at arbitrarily weak attraction; its binding energy is not a power series in g.

In a condensate one instead decouples the reduced interaction with $\Delta = g \sum_k \langle c_{-k\downarrow} c_{k\uparrow} \rangle$. For real positive Δ the pair-creation term has coefficient $-\Delta$. Diagonalizing the particle–hole block gives

$$E_k = \sqrt{\xi_k^2 + \Delta^2}, \qquad |v_k|^2 = (1 - \xi_k/E_k)/2, \qquad |u_k|^2 = 1 - |v_k|^2.$$

The gap equation contains the thermal factor $1 - 2f(E_k)$. With the sum now covering both signs of ξ_k, the weak-coupling gap exponent differs by a factor of two from the isolated Cooper pair exponent. A band cutoff, the number of spin species in the density of states, and the normalization of g must agree before comparing the two.

Case study: Self-consistency from the anomalous average

The thermal anomalous average $F_k = \langle c_{-k\downarrow} c_{k\uparrow} \rangle = \frac{\Delta}{2E_k} \tanh \frac{E_k}{2T}$ gives the gap equation $\Delta = \frac{g}{N_s} \sum_k F_k$. Here it is solved for a band with four momenta, $\xi_k = -2\cos k + 1/2$, and $g = 3$, at $T = 1/10$:

In[1]:=
```
snegfermionoperators[ck];
snegrealconstants[Dl, x, T];
snegrealfunctions[xi];
snegmomentumindexes[{k}, Ns];
hk = First[bdghamiltonian[sum[xi[k] number[ck[k]] - Dl (nc[ck[CR, k,
    UP], ck[CR, -k, DO]] + nc[ck[AN, -k, DO], ck[AN, k, UP]]), {k}],
    {ck}, k]] /. xi[-k] -> xi[k];
F = FullSimplify[ExpToTrig[gaussianaverage[nc[ck[AN, -k, DO], ck[AN,
    k, UP]], hk, {ck}, T, "Momentum" -> k] /. xi[k] -> x], Dl > 0 &&
    T > 0 && Element[x, Reals]]
```

Out[6]=
$$\frac{\mathrm{Dl}\tanh\left(\frac{\sqrt{\mathrm{Dl}^2+x^2}}{2T}\right)}{2\sqrt{\mathrm{Dl}^2+x^2}}$$

In[7]:=
```
FindRoot[Dl == 3/4 Sum[F /. {x -> -2 Cos[q] + 1/2, T -> 1/10}, {q, {0,
     Pi/2, Pi, 3 Pi/2}}], {Dl, 1}]
```

Out[7]= $\{\mathrm{Dl} \to 1.0297\}$

The same gap follows from the self-consistent mean-field solution of the attractive Hubbard ring with `meanfieldsolve`.

11.2 Nambu doubling and bound states

For a single spinful orbital there are two physical positive-energy quasiparticles, not four. The negative eigenvalues of the BdG matrix are their particle–hole partners. A bound state can cross zero as a function of phase or exchange coupling. Its occupation and the many-body ground-state parity then matter even if a plot only shows positive energies. In an interacting dot at large lead gap the induced pairing mixes $|0\rangle$ and $|2\rangle$; the two singly occupied states remain a spin doublet. Comparing the lower even eigenvalue to the odd one locates a crossing without broadening.

Case study: An interacting level with induced pairing

Replace the superconducting continuum by one paired orbital with gap parameter Δ. Unlike a static pair field on the impurity, this model retains a bath quasiparticle. Compare the lowest states with integer and half-integer spin as the repulsion increases, at $\Delta = 1$ and hopping $v = 1/2$:

In[1]:=
```
snegfermionoperators[d,f];
snegrealconstants[U];
hp = U isozsq[d[]]/2 - bcs[f[]] + hop[f[],d[]]/2;
mm = makematricesbzvc[hp,szbasisvc[{d[],f[]}]];
emin[sz_,u_] := Min[Eigenvalues[N[First[keyselect[mm,{sz}]][[2]] /. U
     -> u]]];
rows = Table[{u,emin[0,u],emin[1/2,u]}, {u,0,4}];
TableForm[rows,TableHeadings -> {None,{"U","Lowest Sz = 0","Lowest Sz
     = 1/2"}}]
```

Out[7]=

U	Lowest Sz = 0	Lowest Sz = 1/2
0	−1.41421	−1.20711
1	−1.07067	−1.15139
2	−0.801938	−1.11803
3	−0.610839	−1.09629
4	−0.481194	−1.08114

The ordering reverses between $U = 0$ and $U = 1$. States in the two columns have opposite fermion parity and may cross without mixing. The energies include the constant in the symmetric charging term $U(n_d - 1)^2/2$. For an ideal short single-channel junction with normal transmission τ and equal lead gaps, $E_A = \Delta\sqrt{1 - \tau \sin^2(\phi/2)}$. At zero temperature the even ground branch contributes $I = (2e/\hbar)\partial_\phi(-E_A)$; counting the two Nambu partners as independent filled levels would double this current. A local magnetic exchange can also bind a quasiparticle below the gap. The induced-pairing dot is a useful limiting model, but it does not retain the continuum at $|\omega| = \Delta$ or its gap-edge singularity.

11.2.1 Reading the current from the parity branches

For a dot in the infinite-gap limit, amplitudes for transferring a pair through the two contacts add coherently. In a gauge where the left contact is real, $\Gamma_\phi = \Gamma_L + \Gamma_R e^{i\phi}$. At particle–hole symmetry the lowest even energy is $-|\Gamma_\phi|$; the odd energy is $-U/2$ and has no phase dependence in this limit. The equilibrium current switches branches at a ground-state crossing. Fixing the parity experimentally instead follows one branch through the crossing. Here $\Gamma_L = 1$, $\Gamma_R = 3/5$ and $U = 2$.

In[1]:=
```
ee[p_] := -Sqrt[1+(3/5)^2+2 (3/5) Cos[p]];
eo = -1;
TableForm[Table[{p, N[ee[p]], eo,
  If[ee[p] < eo, N[(3/5) Sin[p]/(-ee[p])], 0]}, {p,{0,Pi/3,2 Pi/3,Pi
    }}],
 TableHeadings -> {None,{"Phase","Even","Odd","Current / I0"}}]
```

Out[3]=

Phase	Even	Odd	Current / I0
0	−1.6	−1	0.
$\frac{\pi}{3}$	−1.4	−1	0.371154
$\frac{2\pi}{3}$	−0.87178	−1	0
π	−0.4	−1	0

The current scale is $I_0 = 2e\Gamma_L/\hbar$. At a crossing the ground energy need not have a derivative; take one-sided derivatives rather than averaging them. A nonzero odd-branch current at finite lead gap is not captured by the static model.

Further reading

For the role of the superconducting continuum in variational states and in Andreev spin qubits see Refs. [9, 10]. The residual quasiparticle in a strongly coupled junction is discussed in Ref. [11].

11.3 Odd-parity pairing and end modes

For a uniform spinless chain the particle–hole diagonal term is $-\mu - 2t\cos k$ and nearest-neighbour odd pairing has magnitude $2|\Delta \sin k|$. A bulk gap closes

at $k = 0$ or π when $|\mu| = 2|t|$ for nonzero Δ. An open chain in the intervening phase has end Majoranas; finite chains generally have an exponentially small splitting, not an exactly protected zero eigenvalue at every length.

Case study: The bulk closing at the topological transition

The bulk gap of the Kitaev chain closes at $k = 0$ for $\mu = -2t$ and at $k = \pi$ for $\mu = 2t$; these are the boundaries of the topological phase $|\mu| < 2t$:

In[1]:=
```
snegspinlessfermionoperators[f, fk];
snegrealconstants[t, mu, Dl];
sneglatticeindexes[{R}, Ns];
snegmomentumindexes[{k}, Ns];
H = sum[-t hop[f[R], f[R + 1]] - mu number[f[R]] + Dl (nc[f[CR, R, 0],
     f[CR, R + 1, 0]] + nc[f[AN, R + 1, 0], f[AN, R, 0]]), {R}];
hk = First[bdghamiltonian[fourier[H, f -> fk, R -> k], {fk}, k]];
e2 = Simplify[ExpToTrig[blochphaseexp[Max[Eigenvalues[hk]]^2, {k -> q
    }]]];
Factor[{e2 /. q -> 0, e2 /. q -> Pi}]
```

Out[8]= $\{(\mu + 2t)^2, (\mu - 2t)^2\}$

11.4 Pair pseudospin on a bipartite lattice

The operator $\eta^+ = \sum_j (-1)^j c^\dagger_{j\uparrow} c^\dagger_{j\downarrow}$ creates an on-site pair with staggered phase. It commutes with nearest neighbour bipartite hopping: the two ends of a bond contribute with opposite signs. The local charge term gives $[H - \mu N, \eta^+] = (U - 2\mu)\eta^+$, so at half filling it generates a charge SU(2) alongside ordinary spin SU(2). This exact symmetry of the Hubbard Hamiltonian is distinct from introducing a BCS anomalous average.

Exercise 11.1 (*The atomic parity crossing*). For one level with repulsion U and induced pairing amplitude $\Gamma > 0$, put $\epsilon = -U/2$. Find the two even energies and the odd energy. At what U/Γ is the lowest state a doublet?

12 Magnetism

12.1 Exchange on a cluster

A positive J in $J\mathbf{S}_i \cdot \mathbf{S}_j$ favours a singlet. Superexchange from a half-filled one-band model has this sign, whereas a direct Hund interaction favours alignment. In double exchange a mobile carrier aligned with local moments instead gains kinetic energy when neighbouring moments are parallel. These mechanisms need not produce the same dependence on filling or on bond angle.

For three spins 1/2 on an equilateral triangle all pair bonds are present, so the energy depends only on the total spin: $H = J[\mathbf{S}_{\mathrm{tot}}^2 - 9/4]/2$. Its two low-spin doublets have the same energy even though no state is a singlet on all three bonds at once. On a square ring not all pairs are neighbours; the Casimir alone no longer fixes each energy.

Case study: Resolving the four-site ring by magnetization

With four spins 1/2 and equal antiferromagnetic bonds, a ring has five magnetization sectors. Diagonalize the blocks at $J = 1$ and tabulate their dimensions and lowest energies:

In[1]:=
```
snegspinoperators[{S,1/2}];
h = Sum[spinspin[S[i],S[Mod[i,4]+1]], {i,4}];
mm = spinclustermatrices[h,{S -> Range[4]}];
rows = {#[[1,1]],Length[Last[#]],Min[Eigenvalues[Last[#]]]} & /@ mm;
TableForm[rows,TableHeadings -> {None,{"Sz","Dimension","Energy / J
    "}}]
```

Out[5]=

Sz	Dimension	Energy / J
−2	1	1
−1	4	−1
0	6	−2
1	4	−1
2	1	1

The minimum $-2J$ occurs only at $S^z = 0$; a higher-spin ground multiplet would also have a representative with nonzero magnetization. The saturated states provide a sign check: each of four parallel bonds has energy $J/4$.

12.2 Spin waves and their regime of validity

Expanding around a ferromagnet polarized along $+z$ gives $S^z = s - b^\dagger b$, $S^+ \simeq \sqrt{2s}\, b$. In $-J \sum_i \mathbf{S}_i \cdot \mathbf{S}_{i+1}$ the quadratic magnon energy is $2Js(1 - \cos k)$; the absent gap follows from the global spin-rotation symmetry. For a Néel reference state rotate one sublattice first. Opposite classical spins then become parallel local axes, and the transverse exchange generates *pairs* of bosons. Diagonalizing the bosonic quadratic form requires the commutation metric with opposite signs on particles and holes; a unitary diagonalization of the Hermitian energy matrix does not give the magnon frequencies.

Case study: A staggered field as a gap regulator

The Heisenberg ring with four spins 1/2 in a staggered field, $H = J \sum_i \mathbf{S}_i \cdot \mathbf{S}_{i+1} - h \sum_i (-1)^{i+1} S^z_i$, expanded with Holstein-Primakoff bosons around the Néel state (sublattice B rotated by π about x). The magnon energies are $\sqrt{(1+h)^2 - \cos^2 k}$, $k = 0, \pm\pi/2, \pi$, in units of J, and E_{gs} includes the zero-point energy:

In[1]:=
```
snegspinoperators[{S, 1/2}];
snegbosonoperators[b];
snegrealconstants[hs];
h = Sum[spinspin[S[i], S[Mod[i, 4] + 1]], {i, 4}] - hs Sum[(-1)^(i +
    1) S[i, 3], {i, 4}];
h = rotatespin[h, S, If[EvenQ[First[{##}]], RotationMatrix[Pi, {1, 0,
     0}], IdentityMatrix[3]] &];
hb = holsteinprimakoff[h, S -> b, 2];
{ε, T, Egs} = bogoliubov[hb /. hs -> 1/10, Table[b[i], {i, 4}]];
{ε, Egs}
```

Out[8]= $\{\{0.458258, 0.458258, 1.1, 1.1\}, -1.84174\}$

In[9]:=
```
N[Sqrt[(1 + 1/10)^2 - Cos[{0, Pi/2, Pi/2, Pi}]^2]]
```

Out[9]= $\{0.458258, 1.1, 1.1, 0.458258\}$

The quadratic truncation assumes a small mean occupation relative to $2s$. A gapless antiferromagnetic chain of spin $1/2$ is particularly unfavourable for a broken-symmetry expansion; finite positive field in the example is part of the Hamiltonian, not just a numerical tolerance.

12.3 One dimension: strings and boundaries

The Jordan–Wigner string counts the occupied sites to the left of each spin flip. Adjacent strings cancel in the bulk. Long-range flips retain intervening density factors, and closing a chain into a ring introduces a boundary sign that depends on total fermion parity. The open XXZ chain is therefore a useful spectrum-level test before handling periodic boundary conditions.

Case study: The interacting spinless chain

XXZ chain of four sites, $H = \sum_i [\frac{J}{2}(S_i^+ S_{i+1}^- + S_i^- S_{i+1}^+) + J_z S_i^z S_{i+1}^z]$. The transverse part gives nearest-neighbor hopping, the longitudinal part a nearest-neighbor interaction:

In[1]:=
```
snegspinoperators[{S, 1/2}];
snegspinlessfermionoperators[f];
snegrealconstants[J, Jz];
H = Sum[J/2 (nc[S[i, 4], S[i + 1, 5]] + nc[S[i, 5], S[i + 1, 4]]) +
    Jz nc[S[i, 3], S[i + 1, 3]], {i, 3}];
Hf = jordanwigner[H, S -> f, Range[4]]
```

Out[5]=
$$\frac{3\mathrm{Jz}}{4} - \frac{\mathrm{Jz} f^\dagger_{1,0} f_{1,0}}{2} + \frac{J f^\dagger_{1,0} f_{2,0}}{2} + \frac{J f^\dagger_{2,0} f_{1,0}}{2} + \mathrm{Jz}(-f^\dagger_{2,0} f_{2,0}) + \frac{J f^\dagger_{2,0} f_{3,0}}{2} + \frac{J f^\dagger_{3,0} f_{2,0}}{2} + \mathrm{Jz}(-f^\dagger_{3,0} f_{3,0}) + \frac{J f^\dagger_{3,0} f_{4,0}}{2} + \frac{J f^\dagger_{4,0} f_{3,0}}{2} - \frac{\mathrm{Jz} f^\dagger_{4,0} f_{4,0}}{2} + \mathrm{Jz}(-f^\dagger_{1,0} f^\dagger_{2,0} f_{1,0} f_{2,0}) + \mathrm{Jz}(-f^\dagger_{2,0} f^\dagger_{3,0} f_{2,0} f_{3,0}) + \mathrm{Jz}(-f^\dagger_{3,0} f^\dagger_{4,0} f_{3,0} f_{4,0})$$

The spectrum agrees with the exact diagonalization of the spin model:

In[6]:=
```
e1 = Sort[Flatten[Eigenvalues /@ makematricesbzop[Hf /. {J -> 1, Jz
    -> 1/2}, qszbasis[Table[f[i], {i, 4}]]][[All, 2]]]];
e2 = Sort[Flatten[Eigenvalues /@ spinclustermatrices[H /. {J -> 1, Jz
     -> 1/2}, {S -> Range[4]}][[All, 2]]]];
Chop[N[e1 - e2]]
```

Out[8]=
$$\{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\}$$

12.4 Itinerant and single-ion magnetism

In a uniform collinear Hartree state, an up electron feels $Un_\downarrow$. Linearizing the spin-density difference gives $m = U\rho(E_F)m$ at zero temperature for a smooth one-spin density of states. A finite cluster needs a thermal derivative rather than a continuum density at a single energy. Passing this uniform instability says nothing about the preferred ordering wave vector.

Case study: A fixed-density susceptibility test

In the Hartree approximation, a paramagnet becomes ferromagnetic when $U\chi_0 > 1$, where $\chi_0 = \frac{1}{N_s}\sum_k[-f'(\epsilon_k)]$ is the Pauli susceptibility per spin and site (Stoner criterion; at $T = 0$, χ_0 is the density of states at the Fermi level). For a Hubbard ring with six sites and four electrons at $T = 0.1$:

In[1]:=
```
L = 6; T0 = 1/10; Ne = 4;
ek = Table[-2 Cos[2 Pi j/L], {j, 0, L - 1}];
mu0 = mu /. FindRoot[2 Total[1/(Exp[(ek - mu)/T0] + 1)] == Ne, {mu,
    -1}];
Uc = 1/(Total[1/(4 T0 Cosh[(ek - mu0)/(2 T0)]^2)]/L)
```

Out[4]= 1.19989

Below U_c the iteration converges to the paramagnetic state, above U_c to a ferromagnetic state with the magnetization per site m:

In[5]:=
```
snegfermionoperators[c];
ring[U_] := -Sum[hop[c[i], c[Mod[i, L] + 1]], {i, L}] + U Sum[hubbard
    [c[i]], {i, L}];
magnetization[U_] := Module[{s = meanfieldsolve[meanfield[ring[U], "
    Hartree"], Table[c[i], {i, L}], Flatten[Table[{avg[number[c[i],
    UP]] -> 0.5, avg[number[c[i], DO]] -> 0.2}, {i, L}]], "
    ParticleNumber" -> Ne, "Temperature" -> T0]},
  Chop[Sum[avg[number[c[i], UP]] - avg[number[c[i], DO]], {i, L}]/L /.
     s["Averages"], 10^-6]];
{magnetization[0.9 Uc], magnetization[1.3 Uc]}
```

Out[8]= $\{0, 0.250777\}$

For a magnetic ion with $s \geq 1$, $D(S^z)^2$ can split a multiplet even in zero field. A term $E[(S^x)^2 - (S^y)^2]$ mixes states whose m_z differs by two. Stevens operators preserve the connection between crystal symmetry and allowed tensor components. For a bond lacking inversion symmetry the antisymmetric exchange $\mathbf{D}_{ij} \cdot (\mathbf{S}_i \times \mathbf{S}_j)$ is allowed; $\mathbf{D}_{ji} = -\mathbf{D}_{ij}$ is required by reversing the bond. It is a two-site coupling, not a single-ion anisotropy.

Case study: Zero-field splitting of a spin one

Zero-field splitting of a spin 1 with axial and rhombic anisotropy, $H = D(S^z)^2 + E[(S^x)^2 - (S^y)^2]$. The levels are 0 and $D \pm E$:

In[1]:=
```
snegspinoperators[{T, 1}];
snegrealconstants[Dz, E2];
H = Dz/3 (stevens[2, 0, T[1]] + 2) + E2 stevens[2, 2, T[1]];
Eigenvalues[spinclustermatrices[H, {T[1]}, "Blocks" -> False]]
```

Out[4]= $\{0, \mathrm{Dz} - \mathrm{E2}, \mathrm{Dz} + \mathrm{E2}\}$

Exercise 12.1 (*A frustrated triangle*). Find the energies and degeneracies of the Heisenberg triangle at $J > 0$. Reconstruct its zero-field partition function; the entropy at $T \to 0$ counts the full manifold, not just highest-weight states.

13 Quantum impurities and nanostructures

13.1 A local level embedded in a band

For one spinful impurity the Anderson Hamiltonian reads

$$H = \sum_{k\sigma} \epsilon_k n_{k\sigma} + \epsilon_d n_d + U D_d + \sum_{k\sigma} (V_k c^\dagger_{k\sigma} d_\sigma + \text{h.c.}).$$

At $V_k = 0$ the local moment exists for $\epsilon_d < 0 < \epsilon_d + U$. A bath adds infinitely many low-energy configurations; it cannot in general be represented by a constant shift of the impurity level. Already at $U = 0$ the retarded hybridization is $\Delta(z) = \sum_k |V_k|^2/(z - \epsilon_k)$, whose imaginary part describes the widths of levels only in the continuum limit.

Case study: Hybridization to a local lattice orbital

Anderson impurity coupled to the site $R = 0$ of the host lattice; the impurity hybridizes with all Bloch states with $V_k = V/\sqrt{N_s}$:

In[1]:=
```
snegfermionoperators[c, ck, d];
snegrealconstants[V];
sneglatticeindexes[{R}, Ns];
snegmomentumindexes[{k}, Ns];
fourier[V hop[d[], c[0]], c -> ck, R -> k]
```

Out[5]=
$$\sum_k \left(\frac{\text{ck}^\dagger_{k\downarrow} d_\downarrow V}{\sqrt{\text{Ns}}} + \frac{\text{ck}^\dagger_{k\uparrow} d_\uparrow V}{\sqrt{\text{Ns}}} + \frac{d^\dagger_\downarrow \text{ck}_{k\downarrow} V}{\sqrt{\text{Ns}}} + \frac{d^\dagger_\uparrow \text{ck}_{k\uparrow} V}{\sqrt{\text{Ns}}} \right)$$

Case study: Frequency-dependent hybridization with one bath level

A level hybridized to one bath orbital has a frequency-dependent self-energy $v^2/(z-e_b)$ even without an interaction:

In[1]:=
```
snegfermionoperators[d, c];
snegrealconstants[ed, eb, v];
h = ed number[d[]] + eb number[c[]] + v hop[d[], c[]];
g = greensfunction[d[AN, UP], d[CR, UP], h, z];
Factor[z - ed - 1/g]
```

Out[5]=
$$\frac{v^2}{z-\mathrm{eb}}$$

13.2 From valence fluctuations to Kondo exchange

For initial and final bath states close to the Fermi level, eliminate empty and doubly occupied impurity states in second order. For real constant V the exchange multiplying $\mathbf{S}_d \cdot \mathbf{s}(0)$ is $J = 2V^2[1/(-\epsilon_d) + 1/(\epsilon_d + U)]$. Both virtual paths favour the impurity–bath singlet. Their *difference* controls potential scattering, which vanishes at $\epsilon_d = -U/2$. The spin density here uses Pauli matrices divided by two; changing that definition changes the quoted J. This elimination is controlled when both charge excitation energies are large on the hybridization scale. It does not directly compute the nonperturbative Kondo scale.

Case study: Checking exchange against exact diagonalization

For a particle-hole symmetric dot and a single bath level at zero energy the half-filled singlet hybridizes with two virtual charge states. Tabulate the energies at $U = 4$, $V = 1/2$ using one row per multiplet. Here $Q = N - 2$ and the energy zero is the empty system:

In[1]:=
```
snegfermionoperators[d,c]; snegrealconstants[U,V];
h = -U number[d[]]/2 + U hubbard[d[]] + V hop[d[],c[]];
b = Select[qsbasisvc[{d[],c[]}], #[[1,1]] == 0 &];
m = makematricesbzvc[h,b];
rows = Flatten[Table[{blockdata[[1,2]],#} & /@
  Sort[Eigenvalues[Last[blockdata] /. {U -> 4,V -> 1/2}]],
  {blockdata,m}],1];
TableForm[rows,TableHeadings -> {None,{"Total spin","Energy"}}]
```

Out[6]=

Total spin	Energy
0	0
0	$-1-\sqrt{2}$
0	$\sqrt{2}-1$
1	-2

The isolated triplet has energy $-U/2$. The lower singlet is $-(U+\sqrt{U^2+64V^2})/4$. Check the leading difference against an antiferromagnetic exchange rather than using the entire singlet energy as the exchange:

In[7]:=
```
gap = -U/2 + (U+Sqrt[U^2+64 V^2])/4;
Limit[gap/V^2, V -> 0, Assumptions -> U > 0]
```

Out[8]=
$$\frac{8}{U}$$

13.3 Wilson chain and NRG Ljubljana

A uniform finite bath has a lowest level spacing that competes with the screening scale. NRG instead divides the band into logarithmic intervals and transforms the retained combinations into a chain. Only the first chain orbital couples to the impurity. Far along a metallic Wilson chain the hopping typically decays as $\Lambda^{-n/2}$ for logarithmic discretization parameter $\Lambda > 1$. Successive diagonalizations can then discard high-energy states relative to the next hopping. Check a discretization and truncation sequence; a two-site singlet in an untruncated test is a normalization check, not evidence by itself of continuum screening.

In the public NRG Ljubljana tree, `test/c++/test42_model_1orb/model.m` and its companion `param` give a complete one-channel initializer example.

`def1ch[1]` sets up the impurity and bath end; the local term uses `number[d[]]` and `hubbard[d[]]`, the contact uses `hop[f[0],d[]]`, and the host term is included once. The initializer derives the contact normalization from the discretized bath; inserting the continuum width Γ as the hopping would have the wrong normalization. The setting `symtype=QS` assumes charge $\mathrm{U}(1)$ and spin $\mathrm{SU}(2)$ invariance; choose different quantum numbers if the model has pairing, spin anisotropy or a magnetic field. Specify measured observables too: an energy spectrum contains no information about matrix elements of a local spin flip.

13.4 Charge stability and superconducting islands

In a double dot the electrostatic term $U_{12} n_1 n_2$ competes with the on-site U; near the $(1,1)$ configuration the virtual charge-transfer cost is $U - U_{12}$ for equivalent sites. The same second-order hopping that mixes $(2,0)$ and $(0,2)$ with a singlet therefore controls the singlet–triplet splitting. Close to a charge degeneracy both charge configurations must be retained. For multi-orbital dots a high-spin atomic multiplet can replace the spin-1/2 moment (see Chapter 10 and Ref. [8]).

A superconducting island brings both pairing and charging energy $E_C(N - N_g)^2$. A static pair field can model proximity to a bulk reservoir but is not a number-conserving description of a floating island. Compare even and odd *total* parity sectors, include the energy of a residual quasiparticle, and distinguish a spin's local response from its total-spin quantum number. The latter distinction enters the impurity Knight shift [12]; crystal fields provide another way of altering the impurity response without changing its charge [13].

Exercise 13.1 (*Where does the pole weight go?*). Hybridize a noninteracting level at zero energy with one bath level also at zero energy by a real $V > 0$. Find both poles of the impurity propagator, their residues per spin, and the limit as $V \to 0$.

14 Fermi liquids, Green's functions and response

14.1 Resolvent or thermal trace?

There are two ways to compute a retarded propagator on a finite cluster. Commuting A with H generates a space of operators. If it closes with $[O_i, H] = \sum_j M_{ij} O_j$, solving a linear system at $z = \omega + i\eta$, $\eta > 0$, sums all nested commutators. Alternatively, a Lehmann sum uses the many-body energies and transition matrix elements. The first route can remain symbolic; the second is a numerical reference that includes correlations without a factorization.

Case study: How large an operator space does a chain require?

For a noninteracting chain, successive commutators propagate the electron to neighbouring sites. No creation or composite operators are generated:

In[1]:=
```
snegfermionoperators[c];
snegrealconstants[t];
h = -t (hop[c[1], c[2]] + hop[c[2], c[3]]);
{ops, m, a} = eomclosure[c[AN, 1, UP], h];
{ops, Factor[CharacteristicPolynomial[m, z]]}
```

Out[5]= $\{\{c_{1\uparrow}, c_{2\uparrow}, c_{3\uparrow}\}, z\left(2t^2 - z^2\right)\}$

Case study: Positive diagonal weight versus signed cross weight

A bonding and an antibonding orbital contribute with opposite signs to propagation from site 2 to site 1. The integrated off-diagonal weight is zero:

In[1]:=
```
snegfermionoperators[c];
h = -hop[c[1], c[2]];
l = lehmann[c[AN, 1, UP], c[CR, 2, UP], h, {c[1], c[2]}, 0];
{l, Chop[Total[l[[All, 2]]]]}
```

Out[4]= $\{\begin{pmatrix} -1. & 0.5 \\ 1. & -0.5 \end{pmatrix}, 0\}$

14.2 High-frequency checks

For an electron the anticommutator fixes the $1/z$ coefficient of the diagonal propagator to one. The next moment probes the mean addition energy, and a higher moment can distinguish a single pole from a pair of poles with the same centroid. For a conserved density the ordinary commutator with itself is zero: a nonzero thermal fluctuation need not produce a nonzero dynamical retarded response.

Case study: An independent sum rule without finding the poles

Verify all four coefficients against a thermal Lehmann sum. For different sites the zeroth moment vanishes, while higher moments can probe hopping:

In[1]:=
```
snegfermionoperators[c];
h = -hop[c[1], c[2]] + 2 Total[hubbard[{c[1], c[2]}]];
modes = {c[1], c[2]};
poles = lehmann[c[AN, 1, UP], c[CR, 2, UP], h, modes, 1/2];
mom = spectralmoments[c[AN, 1, UP], c[CR, 2, UP], h, 3, "Average" ->
    (thermalaverage[#, h, modes, 1/2] &)];
Max[Abs[mom - Table[Total[(Last[#] If[k == 0, 1, First[#]^k]) & /@
    poles], {k, 0, 3}]]] < 10^-9
```

Out[6]= True

14.3 A static limit is a prescription

Define $\chi^R_{AB}(t) = -i\theta(t)\langle[A(t), B]\rangle$. With a perturbation $+hB$, $\delta\langle A\rangle = \chi^R_{AB}h$ in this convention; the common Zeeman coupling is instead $-hS^z$. For the

density of a free band the retarded bubble (spin sum included) is

$$\chi^0(q,z) = \frac{1}{N_s}\sum_{k\sigma}\frac{f(\epsilon_k) - f(\epsilon_{k+q})}{z + \epsilon_k - \epsilon_{k+q}}.$$

At $q = 0$ and finite z the numerator vanishes. Taking the static long-wavelength limit instead gives the negative thermodynamic density of states for a perturbation raising the level energy. In the random phase approximation a repulsive density interaction v_q induces an additional potential $v_q\delta n$, giving $\chi = \chi^0/(1 - v_q\chi^0)$. This argument fixes the denominator sign without importing a convention for a positively defined polarization. A current response to a vector potential also has a diamagnetic term: the current itself depends on the field. Omitting it can produce a spurious response to a pure gauge [2].

14.3.1 A two-level susceptibility by two independent routes

For one electron of fixed spin on a dimer with hopping $-t$, the bonding state has energy $-t$. The imbalance $D = n_1 - n_2$ connects it to the antibonding state at $+t$ with unit matrix element. Thus $\chi^R_{DD}(z) = 1/(z - 2t) - 1/(z + 2t)$. Unlike total charge, this density does not commute with the hopping. Check the static value at $t = 1$ by applying the field directly. In the site basis $+hD$ changes the diagonal energies in opposite directions:

In[1]:=
```
dop = DiagonalMatrix[{1,-1}];
hd[h_] := {{h,-1},{-1,-h}};
{vals,vecs} = Eigensystem[hd[0]];
j = First[Ordering[vals,1]]; l = 3-j;
delta = vals[[l]]-vals[[j]];
wgt = Abs[Conjugate[Normalize[vecs[[j]]]].dop.Normalize[vecs[[l
    ]]]]^2;
TableForm[{{delta,wgt,-2 wgt/delta,D[-Sqrt[1+h^2],{h,2}] /. h -> 0}},
 TableHeadings -> {None,{"Gap","Weight","Retarded limit","Curvature
    "}}]
```

Out[7]=

Gap	Weight	Retarded limit	Curvature
2	1	−1	−1

The last column is the second derivative of the exact lower eigenvalue of

$H + hD$. Its agreement with the retarded limit uses a nondegenerate ground state and a probe independent of field. A thermal mixture also changes its populations, as the exercise below illustrates.

14.4 Quasiparticle parameters and available phase space

Write $G^{-1} = z - \epsilon_k + \mu - \Sigma(k, z)$. For a normal Fermi liquid at a regular point of its Fermi surface, the pole strength is $Z = [1 - \partial_\omega \operatorname{Re} \Sigma(k_F, 0)]^{-1}$. The Fermi velocity is $Z[\nabla_k \epsilon_k + \nabla_k \operatorname{Re} \Sigma(k, 0)]_{k_F}$ in units $\hbar = 1$. Only for a momentum-independent self-energy can the entire mass enhancement of an isotropic parabolic band be read off from $1/Z$. At first order a static Hartree shift changes the Fermi energy but gives no decay. At second order a particle can excite a particle–hole pair; thermal factors restrict all three final quasiparticles to a shell about the Fermi surface. Two independent low-energy integrations remain after energy conservation. With nonsingular phase space in three dimensions the retarded imaginary part is negative and of order $-[\omega^2 + (\pi T)^2]$. In two dimensions logarithms can modify this law. Broadening the discrete lines of a small cluster is not a measurement of that lifetime.

Exercise 14.1 (*Fixed frequency or static field?*). For a single free level of energy ϵ at temperature $T > 0$, find the retarded density commutator and the isothermal derivative $\partial \langle n \rangle / \partial \epsilon$ for one spin. Explain why the two answers differ.

Part III

Function reference

15 Operators, algorithms and interfaces

Entries are grouped by subject. Each gives the usage and details of the corresponding help page and an evaluated example. Short operator identities and selected matrices are printed with their results. See the alphabetical index for symbols used in more than one subject.

15.1 Declaring Operators and Parameters

AN

`AN` is the first index of an annihilation operator: `c[AN, j, ...]` is $c_{j\ldots}$.
`AN` has the value 1, and `CR` the value 0. The values are fixed (`AN` is `Protected`); see `CR` for the consequences.
In the formatted output, annihilation operators are shown without a dagger.
An operator `c[AN, ...]` acting on `VACUUM` gives zero for the default (empty) vacuum. For the Fermi-sea ordering (`ordering[c] = SEA`), `c[AN, k, ...]` with $k < 0$ creates a hole; see `isannihilation`.

Example.

```
snegfermionoperators[c];
{c[AN, 1, UP], AN, nc[c[AN, 1, UP], VACUUM]}
```

coh

`coh[z, c[j]]` gives the normalized coherent state $|z\rangle = e^{-\bar{z}z/2}\, e^{c_j^\dagger z}|0\rangle$ of the mode `c[j]` with the eigenvalue z.
For a fermion mode, z is a Grassmann number declared with `sneggrassmanconstants`. Since $z^2 = 0$, the exponentials terminate:
$|z\rangle = (1 + z\bar{z}/2)|0\rangle - z\, c^\dagger|0\rangle$, with $\bar{z} =$ `conj[z]`.
$|z\rangle$ is an eigenstate of the annihilation operator, $c\,|z\rangle = z\,|z\rangle$, and it is normalized, $\langle z|z\rangle = 1$. The overlap of two coherent states is $\langle w|z\rangle = e^{-\bar{w}w/2 - \bar{z}z/2 + \bar{w}z}$.

The state is written with the placeholder `VACUUM` for $|0\rangle$; `conj[coh[z, c[j]]]` gives the bra $\langle z|$.
The second argument is the mode without the creation/annihilation index: `c[]` for a spinless fermion without indexes, `c[UP]` for the spin-up component of a spin-1/2 fermion `c`, `c[i, UP]` for an orbital index i.
For a boson mode and a complex number z (declared with `snegcomplexconstants`), `coh` gives the bosonic coherent state $e^{-|z|^2/2}e^{za^\dagger}|0\rangle$; the exponential of the operator is kept unexpanded.
Berezin integrals over Grassmann variables are written as `nc[int[z], ...]`, with $\int dz\, z = 1$ and $\int dz\, 1 = 0$.

Example. Fermionic coherent state of a spinless fermion:

```
sneggrassmanconstants[z];
snegspinlessfermionoperators[g];
coh[z, g[]]
```

CR

`CR` is the first index of a creation operator: `c[CR, j, ...]` is $c^\dagger_{j\ldots}$.
`CR` has the value 0, and `AN` the value 1. The values are fixed (`CR` is `Protected`), and several functions depend on them: the normal order with creation operators on the left follows from `OrderedQ` because $0 < 1$, and `conj` maps the first index t to $1 - t$.
Since `CR` evaluates to 0, `FullForm` and `InputForm` show the operators with the index 0; in the formatted output, creation operators are shown with a dagger, $c^\dagger$.
The first index of fermion and boson operators is the type index. Spin operators, Majorana operators, and X-operators have no type index.
For the Fermi-sea ordering (`ordering[c] = SEA`), whether `c[CR, k, ...]` creates or annihilates a quasiparticle depends on the sign of k; see `iscreation`.

Example.

```
snegfermionoperators[c];
{c[CR, 1, UP], CR}
```

DO

`DO` is the spin index of a spin-down fermion: `c[CR, j, DO]` is $c^\dagger_{j\downarrow}$.
`DO` has the value 0, and `UP` the value 1; the values are fixed (`DO` is `Protected`). See `UP` for details.
In the formatted output, `DO` is shown as $\downarrow$.

Example.

```
snegfermionoperators[c];
{c[AN, 2, DO], DO, number[c[2], DO]}
```

halfintegerQ

`halfintegerQ[z]` gives `True` if z is an integer or a half-integer, and `False` otherwise. `halfintegerQ[z]` is `IntegerQ[2 z]`: it is `True` for exact integers and half-integers, positive or negative, and `False` for symbols, approximate numbers, and other rational numbers.
It is used to check spin quantum numbers, e.g. in `spinmatrixX` and `spinbasis`.

Example.

```
{halfintegerQ[3/2], halfintegerQ[-1/2], halfintegerQ[2], halfintegerQ
    [1/3]}
```

int

`nc[int[x], expr]` represents the Berezin integral $\int d\xi$ expr over the Grassmann number $\xi = x$.
`int[x, y, ...]` in a product is `int[x], int[y], ...`.
The Berezin integral is defined by $\int d\xi\, 1 = 0$ and $\int d\xi\, \xi = 1$. In `nc[..., int[x], b, ...]`, the integral acts on the factors to its right: if they do not contain ξ (at the first level of the product), the result is zero; otherwise ξ is moved to the front with the sign of the anticommuting factors it passes (fermions, Majorana fermions, Grassmann numbers), and it is removed together with `int[x]`.
x must be declared with `sneggrassmanconstants`; the conjugate `conj[x]` can also be integrated.
`nc[int[x]]` alone is 0. `int` acts only inside `nc`.
The integrand must be written as a polynomial in the Grassmann numbers: functions such as `Exp` are not expanded. Use the finite Taylor series, e.g. $e^{-a\bar{\xi}\xi} = 1 - a\bar{\xi}\xi$.
The measures `int[...]` are not counted as anticommuting factors. For a single pair of variables this reproduces the standard results, e.g. $\int d\bar{\xi}\, d\xi\, e^{-a\bar{\xi}\xi} = a$; for several variables, see Possible Issues.

Example.

```
sneggrassmanconstants[x, y];
{nc[int[x], 1], nc[int[x], x], nc[int[x], y, x], nc[int[x], y]}
```

listbosonoperators

`listbosonoperators` is the list of the symbols declared as boson operators.

Symbols are added by `snegbosonoperators`. The list is sorted and contains each symbol once.
`listbosonoperators` is a global variable. It is initialized to `{}` when SNEG is loaded, unless it already has a value; it is not used to decide the type of an operator or constant (see the type predicates and `isnumericQ`), so changing it does not change any declaration.

Example.

```
snegbosonoperators[a, b];
listbosonoperators
```

listcomplexconstants

`listcomplexconstants` is the list of the symbols declared as complex constants. Symbols are added by `snegcomplexconstants`. The list is sorted and contains each symbol once.
`listcomplexconstants` is a global variable. It is initialized to `{}` when SNEG is loaded, unless it already has a value; it is not used to decide the type of an operator or constant (see the type predicates and `isnumericQ`), so changing it does not change any declaration.
The lists of constants and `listfreeindexes` are used by `maskOp` to replace the declared constants temporarily by plain symbols.

Example.

```
snegcomplexconstants[z, w];
listcomplexconstants
```

listfermionoperators

`listfermionoperators` is the list of the symbols declared as fermion operators. Symbols are added by `snegfermionoperators` and `snegspinlessfermionoperators`. The list is sorted and contains each symbol once.
`listfermionoperators` is a global variable. It is initialized to `{}` when SNEG is loaded, unless it already has a value; it is not used to decide the type of an operator or constant (see the type predicates and `isnumericQ`), so changing it does not change any declaration.

Example.

```
snegfermionoperators[c, d];
snegspinlessfermionoperators[f];
listfermionoperators
```

listfreeindexes

listfreeindexes is the list of all symbols declared as summation indexes. Symbols are added by snegfreeindexes, sneglatticeindexes, snegmomentumindexes, snegspincomponentindexes, and by the automatic renaming of dummy indexes in products of sums (k1, k2, ...). The list is sorted.
The list is for inspection; the index properties are stored with the symbols.

Example.

```
snegfreeindexes[k, sigma];
sneglatticeindexes[{R}, Ns];
listfreeindexes
```

listgrassmanconstants

listgrassmanconstants is the list of the symbols declared as Grassmann numbers. Symbols are added by sneggrassmanconstants. The list is sorted and contains each symbol once.
listgrassmanconstants is a global variable. It is initialized to {} when SNEG is loaded, unless it already has a value; it is not used to decide the type of an operator or constant (see the type predicates and isnumericQ), so changing it does not change any declaration.

Example.

```
sneggrassmanconstants[x, y];
listgrassmanconstants
```

listintegerconstants

listintegerconstants is the list of the symbols declared as integer constants. Symbols are added by snegintegerconstants. The list is sorted and contains each symbol once.
listintegerconstants is a global variable. It is initialized to {} when SNEG is loaded, unless it already has a value; it is not used to decide the type of an operator or constant (see the type predicates and isnumericQ), so changing it does not change any declaration.
The lists of constants and listfreeindexes are used by maskOp to replace the declared constants temporarily by plain symbols.

Example.

```
snegintegerconstants[n, m];
listintegerconstants
```

listmajoranaoperators

listmajoranaoperators is the list of the symbols declared as Majorana operators. Symbols are added by snegmajoranaoperators. The list is sorted and contains each symbol once.
listmajoranaoperators is a global variable. It is initialized to {} when SNEG is loaded, unless it already has a value; it is not used to decide the type of an operator or constant (see the type predicates and isnumericQ), so changing it does not change any declaration.

Example.

```
snegmajoranaoperators[g, h];
listmajoranaoperators
```

listrealconstants

listrealconstants is the list of the symbols declared as real constants.
Symbols are added by snegrealconstants, and the symbols in symbolic spin sizes (snegspinoperators) and lattice sizes (sneglatticeindexes), which are declared as real constants automatically. The list is sorted and contains each symbol once.
listrealconstants is a global variable. It is initialized to {} when SNEG is loaded, unless it already has a value; it is not used to decide the type of an operator or constant (see the type predicates and isnumericQ), so changing it does not change any declaration.
The lists of constants and listfreeindexes are used by maskOp to replace the declared constants temporarily by plain symbols.

Example.

```
snegrealconstants[t, U];
MemberQ[listrealconstants, #] & /@ {t, U}
```

listspinoperators

listspinoperators is the list of the symbols declared as spin operators.
Symbols are added by snegspinoperators. The list is sorted and contains each symbol once.
listspinoperators is a global variable. It is initialized to {} when SNEG is loaded, unless it already has a value; it is not used to decide the type of an operator or constant (see the type predicates and isnumericQ), so changing it does not change any declaration.

Example.

```
snegspinoperators[{S, 1/2}, T];
MemberQ[listspinoperators, #] & /@ {S, T}
```

listxoperators

`listxoperators` is the list of the symbols declared as Hubbard X-operators. Symbols are added by `snegxoperators`. The list is sorted and contains each symbol once.
`listxoperators` is a global variable. It is initialized to `{}` when SNEG is loaded, unless it already has a value; it is not used to decide the type of an operator or constant (see the type predicates and `isnumericQ`), so changing it does not change any declaration.

Example.

```
snegxoperators[X, "HubbardAtom"];
listxoperators
```

snegbosonoperators

`snegbosonoperators[a, b, ...]` declares a, b, ... as boson operators.
A boson operator `a` is written as `a[t, j, ...]`, where the first index `t` is `CR` for the creation operator $a^\dagger_{j\ldots}$ and `AN` for the annihilation operator $a_{j\ldots}$. The further indexes, if any, label the modes (sites, momenta, polarizations) and can be numbers or symbols. A single mode is written `a[CR]`, `a[AN]`. Boson operators have no spin index.
The operators obey the canonical commutation relations $[a_\alpha, a^\dagger_\beta] = \delta_{\alpha\beta}$, $[a_\alpha, a_\beta] = 0$, where α and β stand for all indexes. Operators of different declared boson families commute with each other, with fermions, and with Majorana fermions.
Products are normal ordered: creation operators to the left of annihilation operators, operators of the same type sorted by their indexes and, for different families, alphabetically. Boson operators are moved to the left of fermion operators.
`conj[a[CR, j]]` is `a[AN, j]`. `number[a[j]]` gives $a^\dagger_j a_j$. `vev` gives vacuum expectation values, and `VACUUM` stands for the boson vacuum $|0\rangle$.
In output, `a[CR, j]` is shown as $a^\dagger_j$.
The declaration sets `operatorQ[a]` and `bosonQ[a]` to `True`, adds `a` to `listbosonoperators`, and gives `a` the attribute `NHoldAll`.
`ordering[a] = NONE` disables the reordering. The commutator is given by `cmt`, which can be redefined for other algebras.
`snegbosonoperators` does not clear existing definitions of the symbols. Use symbols that have no values.

Example.

```
snegbosonoperators[a];
{a[CR], a[AN], a[CR, 1]}
```

snegcomplexconstants

`snegcomplexconstants[z, w, ...]` declares z, w, ... as complex constants.
Declared constants are scalars (see `isnumericQ`): they are factored out of operator products and linear functions.
Complex constants are conjugated by `conj`: `conj[z a]` is `Conjugate[z] conj[a]`.
The declaration sets the upvalue `snegnonopQ[z] = True` and adds z to `listcomplexconstants`.
Use complex constants for complex hopping amplitudes, pairing amplitudes, hybridization with phases, and complex coefficients of states.

Example.

```
snegfermionoperators[c];
snegcomplexconstants[z];
conj[z nc[c[CR, 1, UP], c[AN, 2, UP]]]
```

snegcomplexfunctions

`snegcomplexfunctions[f, g, ...]` declares f, g, ... as complex-valued functions.
After the declaration, every application `f[args]` and its conjugate `Conjugate[f[args]]` are scalars (see `isnumericQ`). `conj[f[args]]` is `Conjugate[f[args]]`.
Typical uses are momentum-dependent hybridizations V_k, form factors, and complex couplings.
The declaration sets `snegnonopQ[f] = True` and `isnumericQ[f[___]] = isnumericQ[Conjugate[f[___]]] = True`. The function symbols are not added to any list.

Example.

```
snegfermionoperators[c, d];
snegfreeindexes[k];
snegcomplexfunctions[V];
conj[V[k] nc[c[CR, k, UP], d[AN, UP]]]
```

snegfermionoperators

`snegfermionoperators[c, d, ...]` declares c, d, ... as spin-1/2 fermion operators.
`snegfermionoperators[{c, s}, ...]` declares c as a fermion operator with spin s.
An operator `c` is written as `c[t, i, j, ..., σ]`. The first index `t` is `CR` for the creation operator $c^\dagger_{ij\ldots\sigma}$ and `AN` for the annihilation operator $c_{ij\ldots\sigma}$. The last index is

the spin projection σ, `UP` or `DO` for spin 1/2; the indexes in between label the orbitals, sites, or momenta and can be numbers or symbols.
The operators obey the canonical anticommutation relations $\{c_\alpha, c_\beta^\dagger\} = \delta_{\alpha\beta}$, $\{c_\alpha, c_\beta\} = 0$, where α and β stand for all indexes. Operators of different declared fermion families anticommute.
Functions that construct operators, such as `number`, `hop`, or `hubbard`, take the level without the first and last indexes: `c[i]` or `c[]`.
For spin s, the spin index takes the values $-s, -s+1, \ldots, s$. Fermions with spin 0 (spinless fermions) are declared with `snegspinlessfermionoperators`.
In output, `c[CR, i, UP]` is shown as $c_{i\uparrow}^\dagger$.
The declaration sets `operatorQ[c]` and `fermionQ[c]` to `True`. All declared fermion operators are listed in `listfermionoperators`.
The products are normal ordered with respect to the vacuum. Set `ordering[c] = SEA` for the particle-hole ordering with respect to the Fermi sea.
The declared symbols get the attribute `NHoldAll`, so that `N` does not change the indexes.
`snegfermionoperators` does not clear existing definitions of the symbols. Declare operators before using them; use symbols that have no values.

Example. Declare `c` and `d` as fermion operators:

```
snegfermionoperators[c, d];
{c[CR, 1, UP], c[AN, 1, UP], d[CR, DO]}
```

snegfreeindexes

`snegfreeindexes[k, q, ...]` declares k, q, ... as indexes that are used as summation variables in `sum[...]` and as operator indexes.
Declared indexes are scalars (see `isnumericQ`): an index appearing in a coefficient, e.g. `k c[CR, k, UP]`, is factored out of products. They are not real: `conj[k]` is `Conjugate[k]`.
The functions for symbolic sums (`sum`, `sumExpand`, `sumSimplifyKD`, `sumCollect`, ...) use the declaration to recognize indexes; when sums are multiplied, dummy indexes that occur in both factors are renamed (`k` to `k1`, `k2`, ...), and the new names are declared automatically.
The declaration sets the upvalues `snegnonopQ[k] = True` and `freeindex[k] = True` and adds k to `listfreeindexes`.
For lattice and momentum sums with periodic boundary conditions, use `sneglatticeindexes` and `snegmomentumindexes`, which also declare the indexes as summation indexes.

Example.

```
snegfermionoperators[c];
snegfreeindexes[k];
nc[sum[c[CR, k, UP], {k}], sum[c[AN, k, UP], {k}]]
```

sneggrassmanconstants

`sneggrassmanconstants[x, y, ...]` declares $x, y, \ldots$ as Grassmann numbers (anticommuting constants).
Grassmann numbers anticommute with each other and with fermion and Majorana operators, $\xi\eta = -\eta\xi$, $\xi c = -c\,\xi$, and $\xi^2 = 0$. They commute with bosons and spin operators.
In products, Grassmann numbers are moved to the left of all operators (with the sign of the permutation) and sorted with `OrderedQ`.
The conjugate `conj[x]` is an independent Grassmann number $\bar{\xi}$, shown as $\bar{x}$. `conj` of products reverses the order: $\overline{\xi\eta} = \bar{\eta}\bar{\xi}$.
Grassmann numbers are not scalars in the sense of `isnumericQ`: they are kept inside `nc[...]` products.
`int[x]` denotes the Berezin integral $\int d\xi$; see `int`.
The declaration sets the upvalues `grassmanQ[x] = True` and `grassmanQ[conj[x]] = True` and adds x to `listgrassmanconstants`.

Example.

```
sneggrassmanconstants[x, y];
{nc[y, x], nc[x, x], conj[x]}
```

snegintegerconstants

`snegintegerconstants[n, m, ...]` declares $n, m, \ldots$ as integer constants.
Integer constants are scalars (see `isnumericQ`) and invariant under complex conjugation, like real constants (`snegrealconstants`).
The declaration sets the upvalues `snegnonopQ[n] = True` and `Conjugate[n] = n`, and adds n to `listintegerconstants`.
The integer property is not used in simplifications: no assumption `n ∈ Integers` is made. Use `Assumptions -> Element[n, Integers]` where needed.

Example.

```
snegfermionoperators[c];
snegintegerconstants[n];
{nc[c[AN, 1, UP], n c[CR, 1, UP]], Conjugate[n], listintegerconstants}
```

snegmajoranaoperators

`snegmajoranaoperators[g, h, ...]` declares g, h, ... as Majorana (real) fermion operators.
A Majorana operator is written as `g[j, ...]` with arbitrary indexes, or as the bare symbol `g` for a single mode. It is Hermitian, $\gamma^\dagger = \gamma$, and there is no creation/annihilation index.
The operators obey $\{\gamma_i, \gamma_j\} = \delta_{ij}$, i.e. $\gamma_i^2 = 1/2$. This is the normalization of $\gamma = (c + c^\dagger)/\sqrt{2}$; with the other common convention $\{\gamma_i, \gamma_j\} = 2\delta_{ij}$, the operators differ by the factor $\sqrt{2}$.
Products are ordered with `OrderedQ` (by indexes, then alphabetically for different families). Majorana operators of different families anticommute. Majorana operators anticommute with Dirac fermions and Grassmann numbers and are moved to the left of Dirac fermions; they commute with bosons.
The declaration sets `operatorQ[g]` and `majoranaQ[g]` to `True`, defines `conj[g] = g`, adds `g` to `listmajoranaoperators`, and gives `g` the attribute `NHoldAll`. The anticommutator is given by `acmt`.
There is no pretty-printing for Majorana operators.

Example.

```
snegmajoranaoperators[g];
{nc[g[1], g[1]], nc[g[2], g[1]], antikomutator[g[1], g[2]], conj[g[1]]}
```

snegrealconstants

`snegrealconstants[x, y, ...]` declares x, y, ... as real constants (parameters).
Declared constants are scalars (see `isnumericQ`): they are factored out of operator products `nc[...]` and of all linear functions, and they commute with all operators.
Real constants are invariant under complex conjugation: `Conjugate[x]` is x, and hence `conj[x a]` is `x conj[a]`.
Typical uses are the parameters of a Hamiltonian, such as hopping t, interaction U, level energy ϵ, exchange J, magnetic field B.
Symbols that are not declared are treated as operators of unknown type: they are not factored out, and the products that contain them are not normal ordered. Every parameter in an operator expression should therefore be declared.
The declaration sets the upvalues `snegnonopQ[x] = True` and `Conjugate[x] = x`, and adds x to `listrealconstants`. The arguments must be symbols; existing values are not cleared.
Assumptions such as $x > 0$ are not implied; use `Assumptions` in `Simplify` or `$Assumptions` when needed.

Example.

```
snegfermionoperators[c];
snegrealconstants[t, U];
nc[c[AN, 1, UP], t c[CR, 2, UP] + U c[CR, 1, UP]]
```

snegrealfunctions

`snegrealfunctions[f, g, ...]` declares *f*, *g*, ... as real-valued functions.
After the declaration, every application `f[args]` is a scalar (see `isnumericQ`), whatever the arguments, and it is real: `Conjugate[f[args]]` is `f[args]`.
Typical uses are dispersion relations ϵ_k, momentum-dependent real couplings V_k, $J(q)$, and distribution functions.
The declaration sets `snegnonopQ[f] = True`, the upvalue `Conjugate[f[args___]] = f[args]`, and `isnumericQ[f[___]] = True`. The function symbols are not added to any list.

Example.

```
snegfermionoperators[c];
snegfreeindexes[k];
snegrealfunctions[ε];
{nc[c[AN, k, UP], ε[k] c[CR, k, UP]], Conjugate[ε[k]]}
```

snegspinlessfermionoperators

`snegspinlessfermionoperators[f, g, ...]` declares *f*, *g*, ... as fermion operators with spin 0.
`snegspinlessfermionoperators[f]` is equivalent to `snegfermionoperators[{f, 0}]`: the operators are fermions with canonical anticommutation relations and `spinof[f]` = 0.
For spin 0, the spin index takes the single value 0. The functions that construct operators from a level, such as `number`, `hop`, `basis`, `qszbasisvc`, `quadraticform`, and `jordanwigner`, therefore write the operators as `f[CR, j, ..., 0]` and `f[AN, j, ..., 0]`, with the spin index 0 as the last index.
In output, the spin index 0 is not shown: `f[CR, 1, 0]` is displayed as $f_1^\dagger$.
Operators can also be written without the spin index, e.g. `f[CR, 1]`; they obey the same algebra, but `f[CR, 1]` and `f[CR, 1, 0]` are different operators. Use one convention consistently; with the functions above, use the trailing 0.
See `snegfermionoperators` for the general properties of fermion operators.

Example.

```
snegspinlessfermionoperators[f];
{spinof[f], number[f[1]], hop[f[1], f[2]]}
```

snegspinoperators

`snegspinoperators[S, ...]` declares S as spin operators that obey the su(2) algebra, without a specified spin size.

`snegspinoperators[{S, s}, ...]` declares S as spin operators of size s, a positive integer or half-integer or a symbolic expression.

A spin operator is written as `S[j, ..., t]`: the indexes `j, ...` label the site (there may be none), and the last index is the component `t` = 1, 2, 3, 4, 5 for S^x, S^y, S^z, S^+, S^-. In output, `S[j, 3]` is shown as S^z_j.

Operators on the same site obey $[S^a, S^b] = i\epsilon_{abc}S^c$, $[S^z, S^\pm] = \pm S^\pm$, $[S^+, S^-] = 2S^z$; operators on different sites commute. For symbolic site indexes, commutators contain $\delta_{jj'}$.

Without a declared size, only these commutation relations are used, which hold in every representation. Products are sorted by site and component ($x < y < z < + < -$), and commutators are applied when operators are exchanged.

With a declared size s, products of operators on the same site are reduced to the normal form $\sum_{a,b} c_{ab}(S^z)^b(S^\pm)^a$:

for every size, $S^+S^- = s(s+1) - (S^z)^2 + S^z$, so that $\mathbf{S}\cdot\mathbf{S} = s(s+1)$, and S^x, S^y in same-site products are written with $S^\pm$;

for numeric s, also the products with $a + b > 2s$ are reduced, e.g. $(S^\pm)^{2s+1} = 0$. The remaining $(2s+1)^2$ monomials are linearly independent, so the normal form is unique and operator identities reduce to zero;

for $s = 1/2$, products of Cartesian components follow the Pauli rule $S^aS^b = \delta_{ab}/4 + (i/2)\epsilon_{abc}S^c$ and stay Cartesian.

A symbolic size, e.g. `snegspinoperators[{S, s}]`, declares the symbols in it as real constants.

Spin operators commute with all other operators. In products they are moved to the left of bosons, Majorana fermions, Dirac fermions, and X-operators; different spin symbols are sorted alphabetically.

`conj` gives $(S^{x,y,z})^\dagger = S^{x,y,z}$ and $(S^\pm)^\dagger = S^\mp$.

The declaration sets `operatorQ[S]` and `spinQ[S]` to `True`, `spinof[S]` to s, adds S to `listspinoperators`, and gives S the attribute `NHoldAll`. A size that is not a positive half-integer or a symbolic expression gives the message `snegspinoperators::size`.

Spin operators of fermion levels, $\mathbf{S} = \frac{1}{2}\sum_{\sigma\sigma'} c^\dagger_\sigma \sigma_{\sigma\sigma'} c_{\sigma'}$, are given by `spinxyz` and related functions; they are operator expressions, not declared spin operators.

Example. Declare spin-1/2 operators:

```
snegspinoperators[{S, 1/2}];
{S[1, 1], S[1, 3], S[1, 4], S[1, 5]}
```

snegxoperators

`snegxoperators[X, {a1, a2, ...}, {p1, p2, ...}]` declares X as Hubbard X-operators for the local states *a1*, *a2*, ... with the fermion parities *p1*, *p2*, ... (0 or 1).
`snegxoperators[X, "HubbardAtom"]` declares X for the four states `"0"`, `"UP"`, `"DO"`, `"2"` of a spin-1/2 orbital.
`X[j, ..., a, b]` is the operator $X_j^{ab} = |a\rangle_j\langle b|$ on the site `j, ...` (there may be no site index); the last two indexes are the states. In output it is shown as X_j^{ab}.
The states carry fermion parities: X^{ab} is a fermionic operator if the parities of a and b differ, and bosonic otherwise.
On the same site, $X^{ab}X^{cd} = \delta_{bc}X^{ad}$. Operators on different sites commute, or anticommute if both are fermionic. For symbolic site indexes, the reordering gives $X_i^{ab}X_j^{cd} = \pm X_j^{cd}X_i^{ab} + \delta_{ij}(\delta_{bc}X^{ad} \mp \delta_{da}X^{cb})$. Products are ordered by site.
$(X^{ab})^\dagger = X^{ba}$.
The completeness relation $\sum_a X^{aa} = 1$ is not applied automatically.
For `"HubbardAtom"`, the states are $|0\rangle$, $|\uparrow\rangle = c_\uparrow^\dagger|0\rangle$, $|\downarrow\rangle = c_\downarrow^\dagger|0\rangle$, $|2\rangle = c_\uparrow^\dagger c_\downarrow^\dagger|0\rangle$, with the parities 0, 1, 1, 0. `xoperator` gives the X-operators in terms of fermions, and `xtofermions` and `fermionstox` convert expressions; `xtospin` maps the operators of singly occupied sites to spin operators.
X-operators are independent of all other operators: they are moved to the right of spin operators and to the left of bosons, Majorana and Dirac fermions, anticommuting with fermions when fermionic. They should therefore not be mixed with fermion operators of the same site.
The declaration sets `operatorQ[X]` and `xoperatorQ[X]` to `True`, adds X to `listxoperators`, and gives X the attribute `NHoldAll`. Invalid arguments (lists of different lengths, repeated states, parities other than 0 and 1) give the message `snegxoperators::spec` and `$Failed`.
With bosonic states only, `snegxoperators[Y, {1, ..., N}, {0, ..., 0}]` gives the generators of SU(N) in the fundamental representation (SU(N) spins).

Example.

```
snegxoperators[X, "HubbardAtom"];
{X[1, "0", "UP"], nc[X[1, "0", "UP"], X[1, "UP", "2"]], nc[X[1, "0", "UP
    "], X[1, "DO", "2"]]}
```

spinof

`spinof[c]` gives the spin s of the fermion operator family c or the spin size of the spin operators c.
For fermions declared with `snegfermionoperators[c]`, `spinof[c]` is 1/2; with `snegfermionoperators[{c, s}]` it is s, and with `snegspinlessfermionoperators[`

`c]` it is 0. The spin index then takes the $2s+1$ values $-s, \ldots, s$ (`DO`, `UP` for $s = 1/2$).
For spin operators declared with `snegspinoperators[{S, s}]`, it is s.
Functions such as `number`, `hubbard`, `basis`, and the basis constructors use `spinof` to generate the spin indexes.
`spinof[x]` is $1/2$ for every x without another value, including undeclared symbols and spin operators declared without size.
The value is an upvalue and can be changed after the declaration with `spinof[c] ^= s`.

Example.

```
snegfermionoperators[c, {f, 3/2}];
snegspinlessfermionoperators[g];
{spinof[c], spinof[f], spinof[g]}
```

UP

`UP` is the spin index of a spin-up fermion: `c[CR, j, UP]` is $c^\dagger_{j\uparrow}$.
`UP` has the value 1, and `DO` the value 0; the values are fixed (`UP` is `Protected`). The spin index is the last index of a fermion operator.
For fermions with spin $s \neq 1/2$ (`snegfermionoperators[{c, s}]`), the spin index takes the values $-s, \ldots, s$; `UP` and `DO` are then the numbers 1 and 0 and are interpreted as such.
In the formatted output, the spin index of spin-1/2 operators is shown as ↑ and ↓.
Since `DO` < `UP`, operators of the same type and orbital are ordered with spin down first.

Example.

```
snegfermionoperators[c];
{c[CR, 1, UP], UP, InputForm[c[CR, 1, UP]]}
```

VACUUM

`VACUUM` denotes the reference ket $|0\rangle$ in an operator product; `conj[VACUUM]` denotes its bra.
Write the ket at the right end of `nc` and its bra at the left end. Their overlap is one.
An annihilator acting on the reference ket gives zero, as does a creator acting to the left on the reference bra.
What annihilates the reference state depends on `ordering`. For a filled Fermi sea a creator below the Fermi surface kills the sea. In the occupation-vector representation use `vacuum[]` and the current `BASIS`.

Example.

```
snegfermionoperators[c];
one = nc[c[CR,UP], VACUUM];
{nc[conj[one],one], nc[conj[one],number[c[]],one]}
```

xoperatorQ

`xoperatorQ[X]` gives `True` if X is a symbol declared with `snegxoperators` or an X-operator `X[...]`, and `False` otherwise.
Unlike the other type predicates (`fermionQ`, `spinQ`, ...), `xoperatorQ` gives `False` for all other expressions.

Example.

```
snegxoperators[X, "HubbardAtom"];
{xoperatorQ[X], xoperatorQ[X[1, "0", "UP"]], xoperatorQ[u]}
```

15.2 Operator Algebra

anticommutator

`anticommutator[a, b]` gives the anticommutator $\{a, b\} = ab + ba$; it is a synonym of `antikomutator`.
`anticommutator[args]` evaluates `antikomutator[args]`. See `antikomutator` for details.

Example. Canonical anticommutation relations:

```
snegfermionoperators[c];
{anticommutator[c[AN, 1, UP], c[CR, 1, UP]], anticommutator[c[AN, 1, UP
    ], c[CR, 1, DO]]}
```

antikomutator

`antikomutator[a, b]` gives the anticommutator $\{a, b\} = ab + ba$.
`antikomutator[a, b]` is `nc[a, b] + nc[b, a]`, with both products brought to the canonical order. `anticommutator` is a synonym.
Fermion operators obey the canonical anticommutation relations $\{c_\alpha, c_\beta^\dagger\} = \delta_{\alpha\beta}$, $\{c_\alpha, c_\beta\} = 0$; Majorana operators $\{\gamma_i, \gamma_j\} = \delta_{ij}$ (SNEG normalization $\gamma^2 = 1/2$); Grassmann numbers anticommute with each other and with fermion operators.
The result is not expanded. Use `Expand` to collect the terms; an anticommutator that is a c-number may only be reduced to it after `Expand`.
`antikomutator` is `Listable`: `antikomutator[{a1, a2}, b]` gives `{antikomutator[a1, b], antikomutator[a2, b]}`.

Symbols that are not declared as operators or constants are treated as operators of unknown type, and the result stays in the form `nc[a, b] + nc[b, a]`.

Example. Canonical anticommutation relations:

In[1]:=
```
snegfermionoperators[c];
{antikomutator[c[AN, 1, UP], c[CR, 1, UP]], antikomutator[c[AN, 1, UP
    ], c[CR, 2, UP]], antikomutator[c[CR, 1, UP], c[CR, 1, DO]]}
```

Out[2]= $\{1, 0, 0\}$

block

`block[expr]` wraps an expression to shield its top-level sum from automatic distribution; `nc[block[expr]]` gives *expr*.
The argument evaluates as usual before wrapping; this is not a holding construct like `HoldForm`. The lowercase `block` has no relation to `Block`, which localizes Wolfram Language values.
The wrapper does not define the statistics of a composite operator. Remove it before evaluating an explicit commutator, matrix element or vacuum average.

Example.
```
snegfermionoperators[c];
wrapped = block[c[CR,UP] + c[AN,UP]];
nc[wrapped]
```

commutator

`commutator[a, b]` gives the commutator $[a, b] = ab - ba$; it is a synonym of `komutator`.
`commutator[args]` evaluates `komutator[args]`. See `komutator` for details.

Example. Commutator of the number operator with a creation operator, $[n, c^\dagger] = c^\dagger$:
```
snegfermionoperators[c];
Expand[commutator[number[c[1]], c[CR, 1, UP]]]
```

conj

`conj[expr]` gives the Hermitian conjugate $expr^\dagger$ of an operator expression *expr*.
`conj` is antilinear and reverses the order of products: $(zAB)^\dagger = z^* B^\dagger A^\dagger$, and `conj[a + b]` is `conj[a] + conj[b]`.
Numbers and declared constants are complex conjugated: real constants (`snegrealconstants`) are unchanged, and complex constants

(`snegcomplexconstants`) give `Conjugate[z]`. Functions declared with `snegrealfunctions` are real.
Fermion and boson operators: $c[\mathrm{CR},\ldots] \leftrightarrow c[\mathrm{AN},\ldots]$, with all other indexes unchanged. Majorana operators are self-adjoint.
Spin operators: S^x, S^y, S^z are self-adjoint and $(S^\pm)^\dagger = S^\mp$. X-operators: $(X^{ab})^\dagger = X^{ba}$.
States: `conj[ket[i]]` is `bra[i]` and vice versa; `conj[VACUUM]` represents the bra $\langle 0|$.
Grassmann numbers z (`sneggrassmanconstants`) are conjugated to the independent Grassmann numbers `conj[z]`.
`conj[conj[a]]` is a. `conj` maps over sums `sum[a, {k}]`, exponentials (`conj[Exp[a]]` is `Exp[conj[a]]`), and rules.
`conj` is `Listable`.
Symbols that are not declared as constants are treated as operators; `conj[u]` is then returned unevaluated. Indexes declared with `snegfreeindexes` are not assumed to be real; `conj[k]` gives `Conjugate[k]`.

Example. Hermitian conjugate of a product with a complex coefficient:

In[1]:=
```
snegfermionoperators[c];
snegcomplexconstants[z];
conj[z nc[c[CR, 1, UP], c[AN, 2, DO]]]
```

Out[3]= $c^\dagger_{2\downarrow} c_{1\uparrow} z^*$

EMPTY

`EMPTY` is the value of `ordering[c]` for the normal order of the fermion operators `c` with respect to the empty vacuum.
`ordering[c] = EMPTY` is the default for every declared fermion family: the reference state is the state without particles, $c_\alpha|0\rangle = 0$ for all α, and products are normal ordered with creation operators to the left. See `ordering`.

Example.
```
snegfermionoperators[c];
{ordering[c], nc[c[AN, 1, UP], c[CR, 1, UP]]}
```

inner

`inner[a, b]` gives the inner product $\sum_i a_i b_i$ of the lists a and b, with the products formed by `nc`.
`inner[a, b, g]` uses the function g instead of `Plus` to combine the products.

`inner[a, b, ...]` is `Inner[nc, a, b, ...]`. The order of the factors in each product is preserved, as needed for operators; for operator vectors **A** and **B**, `inner[A, B]` is $\mathbf{A}\cdot\mathbf{B}=\sum_a A^a B^a$.
As for `Inner`, a and b can be tensors: the last index of a is contracted with the first index of b, e.g. for a matrix and a vector.
No complex conjugation is performed. For the scalar product of states, use `scalarproductvc` or `scalarproductop`.
The result is not expanded.

Example.

```
inner[{a1, a2}, {b1, b2}]
```

komutator

`komutator[a, b]` gives the commutator $[a,b]=ab-ba$.
`komutator[a, b]` is `nc[a, b] - nc[b, a]`, with both products brought to the canonical order. `commutator` is a synonym.
The result is not expanded. Use `Expand` to collect the terms; a commutator that vanishes is only reduced to 0 after `Expand`.
Spin operators obey the su(2) algebra $[S_i^a,S_j^b]=i\delta_{ij}\epsilon_{abc}S_i^c$. For symbolic Cartesian components declared with `snegspincomponentindexes`, the commutator is given directly in terms of `levicivita`. Operators of different spin symbols commute.
Commutators of sums `sum[...]` are computed with automatic renaming of dummy indexes. Use `sumExpand` and `sumSimplifyKD` to perform the sums over Kronecker deltas.
`komutator` is `Listable`: `komutator[{a1, a2}, b]` gives `{komutator[a1, b], komutator[a2, b]}`.
The time evolution of an operator in the Heisenberg picture is $i\hbar\dot{A}=[A,H]$; `liouvillian` gives iterated commutators $[[A,H],H],\ldots$.

Example.

In[1]:=
```
snegfermionoperators[c];
komutator[c[CR, 1, UP], c[AN, 2, UP]]
```

Out[2]= $2c^\dagger_{1\uparrow}c_{2\uparrow}$

mcross

`mcross[a, b]` gives the cross product $\mathbf{a}\times\mathbf{b}$ of three-component lists a and b, with the products formed by `nc`.

The components are $(\mathbf{a} \times \mathbf{b})_x = a_y b_z - a_z b_y$, and cyclic. The order of the factors is preserved, so $\mathbf{a} \times \mathbf{b} \neq -\mathbf{b} \times \mathbf{a}$ for noncommuting components.
Only the first three elements of a and b are used.
The su(2) commutation relations $[S^a, S^b] = i\epsilon_{abc} S^c$ are equivalent to $\mathbf{S} \times \mathbf{S} = i\mathbf{S}$.

Example.

```
mcross[{a1, a2, a3}, {b1, b2, b3}]
```

nc

`nc[a, b, c, ...]` represents the noncommutative product $abc\cdots$ of the operators a, b, c, ...
`nc[]` is 1, and `nc[a]` is a.
`nc` is the product of operators. The ordinary product `Times` is commutative and must only be used to multiply operators by scalars.
`nc[a, b]` can also be entered as `a ~nc~ b`. In output, products are shown as $a \cdot b$.
`nc` is linear in each argument: it distributes over sums, and scalar factors are moved to the front. Scalars are numbers, the constants and functions declared with `snegrealconstants`, `snegcomplexconstants`, `snegrealfunctions`, ..., and expressions built from them; see `isnumericQ`.
`nc` is associative: nested products are flattened, `nc[a, nc[b, c]]` is `nc[a, b, c]`.
Products of declared operators are brought to a canonical order. For fermions and bosons this is the normal order: creation operators to the left of annihilation operators, and operators of the same kind sorted by their indexes. Every exchange of two fermion operators gives a factor -1; exchanging an annihilation and a creation operator also gives the contraction term from the canonical (anti)commutation relations $\{c_i, c_j^\dagger\} = \delta_{ij}$ and $[a_i, a_j^\dagger] = \delta_{ij}$.
The canonical order is set for each operator family by `ordering`: `EMPTY` (the default) gives the normal order with respect to the vacuum, `SEA` with respect to the Fermi sea, and `NONE` leaves the products unordered.
A product with two equal fermion operators vanishes (Pauli principle).
Operators of different kinds are reordered as follows: bosons commute with fermions and are moved to the left, spin operators commute with all other operators and are moved to the left, and fermion operators of different families anticommute.
Spin operators with a declared spin size (`snegspinoperators[{S, s}]`) are reduced to a normal form in S^z and $S^\pm$ on each site.
Products can end with a ket (`ket`, `VACUUM`) or start with a bra; annihilation operators acting on `VACUUM` give zero.
Products of sums `sum[...]` are combined into a single sum; dummy indexes that occur in both factors are renamed.
`nc` is `Listable`.

Symbols that are not declared as constants or operators are treated as noncommuting operators of unknown type: they are kept in place and not reordered.
Use `pow[a, n]` for powers of operators: `a^n` is the commutative power.

Example. Declare `c` as a fermion operator. The first index distinguishes creation (`CR`) from annihilation (`AN`) operators, and the last is the spin (`UP` or `DO`):

In[1]:=
```
snegfermionoperators[c];
nc[c[CR, 1, UP], c[AN, 2, UP]]
```

Out[2]= $c^{\dagger}_{1\uparrow} c_{2\uparrow}$

NONE

`NONE` is the value of `ordering[c]` that disables the automatic reordering of products of the operators `c`.
With `ordering[c] = NONE`, products are kept in the order in which they are entered; operators of other families are not exchanged with `c` either. The Pauli principle and linearity still apply. It can be set for fermions and bosons. Expectation values of such products are evaluated by `vevwick`. See `ordering`.

Example.
```
snegfermionoperators[c];
ordering[c] = NONE;
{nc[c[AN, 1, UP], c[CR, 1, UP]], vevwick[nc[c[AN, 1, UP], c[CR, 1, UP
    ]]]}
```

normalorder

`normalorder[expr]` gives `expr - vev[expr]`, the operator *expr* with its vacuum expectation value subtracted.
For a product of two operators, `normalorder` gives the normal-ordered product with respect to the vacuum set by `ordering`, $:AB: = AB - \langle AB \rangle$. For the Fermi sea (`ordering[c] = SEA`), $:c_k^{\dagger} c_k: = c_k^{\dagger} c_k - \theta(-k)$.
For longer products, $x - \langle x \rangle$ is not the normal-ordered product: Wick's theorem also subtracts the partial contractions. Use `wick` to obtain $:x:$ for such products.
The vacuum expectation value is computed by `vev`. For the Fermi sea, `vev` does not evaluate products of more than two operators with symbolic indexes; `normalorderwick` uses `vevwick` instead.
`normalorder` is `Listable`.

For the empty vacuum, `nc` already brings products to normal order; `normalorder` then subtracts the constant term.

Example. Normal ordering with respect to the Fermi sea, $:c_k^\dagger c_k: = c_k^\dagger c_k - \theta(-k)$:

```
snegfermionoperators[c];
ordering[c] = SEA;
snegfreeindexes[k];
normalorder[nc[c[CR, k, UP], c[AN, k, UP]]]
```

ordering

`ordering[c] = EMPTY` selects the normal order of the fermion operators `c` with respect to the empty vacuum (default).
`ordering[c] = SEA` selects the normal order with respect to the Fermi sea, in which the levels with negative momentum index are filled.
`ordering[c] = NONE` disables the automatic reordering of products of the operators `c`.
`ordering[c]` gives the current setting.
`nc` brings products of operators to a canonical order: quasiparticle creation operators to the left of quasiparticle annihilation operators, where an annihilation operator is one that gives zero when it acts on the reference state ("vacuum") $|0\rangle$.
`ordering[c]` defines this reference state for the operator family `c`, and hence which of the operators `c[CR, ...]` and `c[AN, ...]` play the role of creation and annihilation operators.
Every declared fermion operator has `ordering[c] = EMPTY` by default. The setting is an ordinary definition and can be changed at any time; it affects the products evaluated afterwards.
EMPTY: the reference state is the empty state, $c_\alpha|0\rangle = 0$ for all α. Products are ordered with `OrderedQ` applied to the operators themselves. Since `CR` $= 0 <$ `AN` $= 1$, this puts $c^\dagger$ to the left of c, and sorts operators of the same type by their remaining indexes (orbital, then spin).
SEA: the reference state is the Fermi sea $|\mathrm{FS}\rangle$. The second index of an operator `c[t, k, ...]`, the first one after `CR`/`AN`, is the momentum or energy k measured from the Fermi level: the levels with $k > 0$ are empty and those with $k < 0$ are filled. The quasiparticle operators are:

$k > 0$ (particle)	$k < 0$ (hole)
creation $c_k^\dagger$	c_k
annihilation c_k	$c_k^\dagger$

For SEA operators, `iscreation` and `isannihilation` evaluate $k > 0$ and $k < 0$. If the sign of k is not known (a symbol without assumptions, or $k = 0$), the role is undetermined and the EMPTY rule (sorting by the raw indexes) is used as fallback;

the result is then correct but not normal ordered with respect to the Fermi sea. The sign of a symbolic index is declared with an upvalue, e.g. `k /: k > 0 = True`.

For SEA operators, `vev` gives the Fermi-sea expectation values $\langle c_k^\dagger c_q \rangle = \delta_{kq}\,\theta(-k)$ and $\langle c_k c_q^\dagger \rangle = \delta_{kq}\,\theta(k)$, written with `UnitStep` (note `UnitStep[0]` $= 1$). Longer products are evaluated by `vevwick`, which applies Wick's theorem with these contractions. `VACUUM` stands for $|\mathrm{FS}\rangle$ in products with SEA operators.

NONE: products of `c` are kept in the order in which they are entered. The Pauli rule $c_\alpha c_\alpha = 0$ and linearity still apply. A NONE operator also acts as a barrier: operators of other families are not exchanged with it. `vevwick` and `contraction` evaluate expectation values of NONE products (with the contractions from `acmt` and `cmt`); `vev` and the `VACUUM` rules do not. NONE is useful for very long operator strings or when a specific operator order is required.

Bosons have no `ordering` value by default and are normal ordered with respect to the boson vacuum; `ordering[a] = NONE` disables the reordering of the boson operators `a`. SEA is not supported for bosons.

Products of operators of different fermion families are ordered by the roles (`iscreation`, `isannihilation`) that each operator has in its own family, so that families with different settings can be mixed; operators of the same role are sorted with `OrderedQ`, which puts the families in alphabetical order. The exchange gives a factor -1 and no contraction term.

Other values than EMPTY, SEA, and NONE can be used to define custom reference states; see Developer Notes and the example under Applications.

Example. The default is the empty vacuum. A product $c_k c_k^\dagger$ is reordered:

```
snegfermionoperators[c];
{ordering[c], nc[c[AN, 1, UP], c[CR, 1, UP]]}
```

outer

`outer[a, b]` gives the outer product of the lists a and b, the matrix of all products `nc[ai, bj]`.

`outer[a, b, n]` treats only the first n levels of a and b as lists, as in `Outer`.

`outer[a, b, ...]` is `Outer[nc, a, b, ...]`. The factors keep their order, as needed for operators.

`Flatten[outer[a, b]]` gives all products of the elements of a and b; this is used to form product bases, e.g. of fermion states and phonon or spin kets.

Example.

```
outer[{a1, a2}, {b1, b2}]
```

pow

`pow[a, n]` gives the n-th power $a^n = a \cdot a \cdots a$ of the operator expression a, computed with the noncommutative product `nc`.
`pow[a, 0]` is 1, `pow[a, 1]` is a, and `pow[a, n]` is `nc[pow[a, n - 1], a]` for integers $n > 1$.
`Power (a^n)` is the commutative power and gives wrong results for operators; use `pow`.
For negative n, the message `pow::negativepower` is issued and `Null` is returned. For non-integer n, `pow` is returned unevaluated.
The result is not expanded; use `Expand` to collect the terms.
`pow` is `Listable`: for a list a, the powers of the elements are returned.

Example. $(c + c^\dagger)^2 = 1$ for a fermion:

In[1]:=
```
snegfermionoperators[c];
pow[c[AN, UP] + c[CR, UP], 2]
```

Out[2]= 1

PrettyInput

`PrettyInput` is a global setting: if `True` (the default), the center dot `a·b` (`CenterDot`) is interpreted as the operator product `nc[a, b]`.
With `PrettyInput = True`, SNEG defines `CenterDot[a, b, ...] := nc[a, b, ...]` when it is loaded, so that output in the format of `PrettyOutput` can be copied and used as input. The center dot is entered as `Esc . Esc`.
To disable the definition, set Sneg`PrettyInput = False` before loading the package. Changing 'PrettyInput' after loading has no effect.

Example.
```
snegfermionoperators[c];
CenterDot[c[AN, 1, UP], c[CR, 1, UP]]
```

SEA

`SEA` is the value of `ordering[c]` for the normal order of the fermion operators `c` with respect to the Fermi sea.
With `ordering[c] = SEA`, the second index of `c[t, k, ...]` is the momentum or energy measured from the Fermi level: the levels with $k < 0$ are filled and those with $k > 0$ empty in the reference state $|\mathrm{FS}\rangle$. Then $c_k^\dagger$ $(k > 0)$ and c_k $(k < 0)$ are quasiparticle creation operators, and products are normal ordered with respect to $|\mathrm{FS}\rangle$. See `ordering` for details and examples.

Example.

```
snegfermionoperators[d];
ordering[d] = SEA;
{nc[d[CR, -1, UP], d[AN, -1, UP]], vev[nc[d[CR, -1, UP], d[AN, -1, UP
    ]]], vev[nc[d[CR, 1, UP], d[AN, 1, UP]]]}
```

snegAssuming

`snegAssuming[s /: lhs = rhs, expr]` evaluates *expr* with the temporary upvalue `s /: lhs = rhs`.
`snegAssuming[{s1 /: lhs1 = rhs1, ...}, expr]` evaluates *expr* with several temporary upvalues.
The assumptions are given as `TagSet` definitions, typically for the sign of a symbolic index: `k /: k > 0 = True`. They are defined before *expr* is evaluated, and all upvalues and downvalues of the tagged symbols are restored afterwards, also those defined during the evaluation of *expr*.
`snegAssuming` has the attribute `HoldAllComplete`: *expr* is only evaluated after the assumptions are set.
The assumption only affects tests that match the defined left-hand side literally. `k /: k > 0 = True` makes `k > 0` evaluate to `True`, but not `k < 0`, `k >= 0`, `Positive[k]`, or `UnitStep[k]`.
In SNEG, the main use is to decide whether an operator with `ordering[c] = SEA` creates or annihilates an excitation of the Fermi sea: `iscreation` and `isannihilation` test `k > 0` and `k < 0` for the second index *k*.
`Assuming` and `$Assumptions` do not influence these tests, since they are not used by the evaluation of `k > 0`.

Example.

```
snegfreeindexes[k];
{k > 0, snegAssuming[k /: k > 0 = True, k > 0], k > 0}
```

snegSeries

`snegSeries[f, x, n]` gives the Taylor series of $f(x)$ around $x = 0$ to order n, with the powers of the operator expression x computed with `nc`.
`snegSeries[f[x], n]` is `snegSeries[f, x, n]`.
`snegSeries[f, x]` and `snegSeries[f[x]]` use the order 5.
The result is $\sum_{m=0}^{n} \frac{f^{(m)}(0)}{m!} x^m$, where x^m is the noncommutative power. f is a function such as `Exp` or `Sin`, or a pure function such as `Sqrt[1 + #] &`.
The coefficients are computed with `Series`. The powers are computed by repeated `nc` multiplication, with `Simplify` applied at every step.

The series is truncated at order n; products of truncated series contain terms of higher order that are incomplete.
`snegSeries[Exp[x], n]` and `snegSeries[E^x, n]` give the series of the exponential.

Example. The exponential of a boson creation operator:

```
snegbosonoperators[a];
snegrealconstantsλ[];
snegSeries[Exp, λ a[CR], 3]
```

SnegSimplify

`SnegSimplify[expr]` rewrites the operator expression *expr* in terms of the SNEG functions `number`, `hubbard`, `hop`, `twohop`, `spinx`, `spiny`, `spinz`, `spinspin`, and `chargecharge`, wherever their expanded forms are recognized.
`SnegSimplify` makes expanded operator expressions readable. The recognized terms are returned wrapped in `HoldForm`, e.g. `HoldForm[hubbard[c[1]]]`, so that they are not expanded again; `ReleaseHold` gives back the operator expression.
The expression is first collected with respect to the declared constants; the number operators, Hubbard terms, hopping terms, spin operators and spin-spin products, and density-density terms are then replaced in this order.
The rules recognize the terms of fermion operators with spin 1/2 in the form produced by `nc` (normal-ordered, expanded). Terms that do not match are left unchanged.
`SnegFullSimplify[expr]` uses the same replacements as transformation functions of `Simplify` and searches for the simplest combination. It finds more compact forms, e.g. $\mathbf{S}_1 \cdot \mathbf{S}_2 - n_1 n_2/4$, but is slower.
Number operators of the same level are replaced first, so that S^z is given as $(n_\uparrow - n_\downarrow)/2$.

Example. The Hubbard dimer in expanded form:

```
snegfermionoperators[c];
snegrealconstants[t, U, ε];
H = -t hop[c[1], c[2]] + U (hubbard[c[1]] + hubbard[c[2]]) + ε (number[c
    [1]] + number[c[2]])
```

superanticommutator

`superanticommutator[a, b, n]` gives the n-fold nested anticommutator $\{a, \{a, \ldots \{a, b\} \ldots\}\}$; it is a synonym of `superantikomutator`.
`superanticommutator[args]` evaluates `superantikomutator[args]`. See `superantikomutator` for details.

Example.

```
snegfermionoperators[c];
Table[superanticommutator[c[AN, UP], c[CR, UP], k], {k, 0, 3}]
```

superantikomutator

`superantikomutator[a, b, n]` gives the n-fold nested anticommutator $\{a, \{a, \ldots \{a, b\} \ldots\}\}$.
`superantikomutator[a, b, 0]` is b, and `superantikomutator[a, b, n]` is `antikomutator[a, superantikomutator[a, b, n - 1]]`. `superanticommutator` is a synonym.
`Simplify` is applied to the result of every step.
n must be a non-negative integer; otherwise the expression is returned unevaluated.
See `superkomutator` for nested commutators.

Example. Nested anticommutators of $c_\uparrow$ with $c_\uparrow^\dagger$, $\{c, c^\dagger\} = 1$ and $\{c, 1\} = 2c$:

```
snegfermionoperators[c];
Table[superantikomutator[c[AN, UP], c[CR, UP], n], {n, 0, 3}]
```

supercommutator

`supercommutator[a, b, n]` gives the n-fold nested commutator $[a, [a, \ldots [a, b] \ldots]]$; it is a synonym of `superkomutator`.
`supercommutator[args]` evaluates `superkomutator[args]`. See `superkomutator` for details.

Example. Nested commutators with the number operator, $[n, c] = -c$:

```
snegfermionoperators[c];
Table[supercommutator[number[c[]], c[AN, UP], k], {k, 0, 3}]
```

superkomutator

`superkomutator[a, b, n]` gives the n-fold nested commutator $[a, [a, \ldots [a, b] \ldots]]$.
`superkomutator[a, b, 0]` is b, and `superkomutator[a, b, n]` is `komutator[a, superkomutator[a, b, n - 1]]`. `supercommutator` is a synonym.
`Simplify` is applied to the result of every step.
n must be a non-negative integer; otherwise the expression is returned unevaluated.
The nested commutators are the terms of the Hadamard (Baker-Campbell-Hausdorff) expansion of a unitary transformation,

$$e^A B e^{-A} = \sum_{n=0}^{\infty} \frac{1}{n!} [A, [A, \ldots [A, B]]]_n.$$

With $A = iHt$, they give the Taylor expansion of the Heisenberg operator $B(t) = e^{iHt}Be^{-iHt}$. `liouvillian` gives the list of commutators $[B,H],[[B,H],H],\ldots$ with the opposite order of the arguments.

Example. Nested commutators of the spin projection S^z with $c^\dagger_\uparrow$, $[S^z, c^\dagger_\uparrow] = c^\dagger_\uparrow/2$:

```
snegfermionoperators[c];
Table[superkomutator[spinz[c[]], c[CR, UP], n], {n, 0, 3}]
```

VMV

`VMV[v1, m, v2]` gives the product $\sum_{ij}(v_1)_i\, m_{ij}\, (v_2)_j$ of the vectors *v1*, *v2* and the matrix *m*, with the products formed by `nc`.
`VMV[v1, m, v2]` is `inner[inner[v1, m], v2]`. The operator order $v_{1,i}$, $v_{2,j}$ is preserved.
With $v_1 = (c^\dagger_1, c^\dagger_2, \ldots)$ and $v_2 = (c_1, c_2, \ldots)$, `VMV[v1, h, v2]` is the bilinear form $\sum_{ij} c^\dagger_i h_{ij} c_j$, e.g. a tight-binding Hamiltonian or the spin density $\frac{1}{2}\sum_{\sigma\sigma'} c^\dagger_\sigma \sigma_{\sigma\sigma'} c_{\sigma'}$.
More generally, the last index of *v1* is contracted with the first index of *m*, and the last index of *m* with the first index of *v2*. *m* can thus be a list of matrices with *v1* a list of operators, as in $\sum_a \psi_a \sigma^a$.

Example.

```
VMV[{a1, a2}, {{m11, m12}, {m21, m22}}, {b1, b2}]
```

15.3 Building Hamiltonians

anhop

`anhop[a[i], b[j]]` gives the singlet anomalous hopping operator $a^\dagger_{i\uparrow} b^\dagger_{j\downarrow} - a^\dagger_{i\downarrow} b^\dagger_{j\uparrow} + \text{h.c.}$
`anhop[a[i], b[j], σ]` gives $a^\dagger_{i\sigma} b^\dagger_{j\bar\sigma} + b_{j\bar\sigma} a_{i\sigma'}$, the same as `anomaloushop[a[i], b[j], σ]`.
`anhop[fn1, fn2]` gives the singlet anomalous hopping between abstract orbitals given as pure functions.
The levels are spin-1/2 fermion levels given without the creation/annihilation and spin indexes; $\bar\sigma$ is the spin opposite to σ.
`anhop[a, b]` is `anhop[a, b, UP] - anhop[a, b, DO]`. The pair $a^\dagger_\uparrow b^\dagger_\downarrow - a^\dagger_\downarrow b^\dagger_\uparrow$ is a spin singlet, so `anhop` commutes with all components of the total spin. It differs from `anomaloushop` (triplet, $S_z = 0$) by the sign of the $\sigma =\downarrow$ term.
`anhop[a, b]` is symmetric in its arguments, and `anhop[a, a]` is `2 bcs[a]`.
With a complex amplitude, use `genanhop[t, a, b]`; `anhop[a, b]` is `genanhop[1, a, b]`.

Abstract orbitals: `fn` is a pure function of the type and spin indexes such that `fn[CR, σ]` is the creation operator of the orbital. In the two-argument form, one argument may be an ordinary level.
`anhop` is `Listable`.

Example.
```
snegfermionoperators[a, b];
anhop[a[], b[]]
```

anomaloushop

`anomaloushop[a[i], b[j]]` gives the anomalous hopping operator $\sum_\sigma (a^\dagger_{i\sigma} b^\dagger_{j\bar\sigma} + b_{j\bar\sigma} a_{i\sigma}) = a^\dagger_{i\uparrow} b^\dagger_{j\downarrow} + a^\dagger_{i\downarrow} b^\dagger_{j\uparrow} + \text{h.c.}$
`anomaloushop[a[i], b[j], σ]` gives $a^\dagger_{i\sigma} b^\dagger_{j\bar\sigma} + b_{j\bar\sigma} a_{i\sigma}$.
`anomaloushop[fn1, fn2]` gives the anomalous hopping between abstract orbitals given as pure functions.
The levels are spin-1/2 fermion levels given without the creation/annihilation and spin indexes; $\bar\sigma$ is the spin opposite to σ.
The pair $a^\dagger_\uparrow b^\dagger_\downarrow + a^\dagger_\downarrow b^\dagger_\uparrow$ created by `anomaloushop` is the $S_z = 0$ component of a spin triplet. The operator commutes with the total S^z but not with S^x, S^y. For singlet pairing $a^\dagger_\uparrow b^\dagger_\downarrow - a^\dagger_\downarrow b^\dagger_\uparrow + \text{h.c.}$, which conserves the total spin, use `anhop`.
`anomaloushop[a[i], a[i]]` vanishes. Local (on-site) pairing is given by `bcs`.
`anomaloushop[p, h]` equals `holehop[h, p]`: it is the hopping between a particle level `p` and a level `h` described by hole operators.
Abstract orbitals: `fn` is a pure function of the type and spin indexes such that `fn[CR, σ]` is the creation operator of the orbital. In the two-argument form, one argument may be an ordinary level.
`anomaloushop` is `Listable`.

Example.
```
snegfermionoperators[a, b];
anomaloushop[a[], b[]]
```

bcs

`bcs[c[i]]` gives the local pairing operator $c^\dagger_{i\uparrow} c^\dagger_{i\downarrow} + c_{i\downarrow} c_{i\uparrow}$.
`bcs[c[i], Φ]` gives $e^{i\Phi} c^\dagger_{i\uparrow} c^\dagger_{i\downarrow} + e^{-i\Phi} c_{i\downarrow} c_{i\uparrow}$ with the phase *Φ*.
`bcs[Δ, c[i]]` gives $\Delta\, c^\dagger_{i\uparrow} c^\dagger_{i\downarrow} + \Delta^*\, c_{i\downarrow} c_{i\uparrow}$ with the complex amplitude *Δ*.
The level is a spin-1/2 fermion level given without the creation/annihilation and spin indexes. The order of the arguments distinguishes the two forms: a level as the

first argument means `bcs[c[i], Φ]`, a level as the second argument means `bcs[Δ, c[i]]`.
The pair creation operator is $c_\uparrow^\dagger c_\downarrow^\dagger$, with the spin-up operator on the left. The doubly occupied state used in this convention is $|2\rangle = c_\uparrow^\dagger c_\downarrow^\dagger |0\rangle$.
The mean-field (BCS) pairing term of a superconductor with the gap Δ and the phase Φ in a local (Wilson-chain or zero-bandwidth) representation is $-\Delta \sum_i$ `bcs[c[i], Φ]`. For a single level $H = \xi\,(n-1) - \Delta\,\mathrm{bcs}(c, \Phi)$, the even states $|0\rangle$, $|2\rangle$ are mixed into the energies $\pm\sqrt{\xi^2 + \Delta^2}$, and the singly occupied states have the energy 0: the quasiparticle excitation energy is $E = \sqrt{\xi^2 + \Delta^2}$. The ground state for $\xi = 0$ is $(|0\rangle + e^{i\Phi}|2\rangle)/\sqrt{2}$, see `projectorEphi`.
`bcs[c[i], Φ]` is $2(\cos\Phi\, I^x - \sin\Phi\, I^y)$ in terms of the isospin operators of the level (`isospinx`, `isospiny`); `bcs[c[i]]` is $2I^x$. It is a spin singlet: it commutes with the spin operators of the level.
Under the gauge transformation $c_\sigma \to e^{i\chi} c_\sigma$, `bcs[c, Φ]` becomes `bcs[c, Φ - 2χ]`, and a hopping term `hop[c, d]` becomes `hopphi[c, d, -χ]`. The superconducting phase can therefore be moved to the tunneling terms, see `hopphi`.
`bcs[c[i], Φ]` is `bcs[Exp[I Φ], c[i]]`. Nonlocal pairing between two levels is given by `anhop` (singlet) and `genanhop`; `anhop[c[i], c[i]]` is `2 bcs[c[i]]`.
`bcs` is not `Listable`, and bare symbols are not accepted as levels.

Example. The local pairing operator:

In[1]:=
```
snegfermionoperators[c];
bcs[c[]]
```

Out[2]= $-c_\downarrow^\dagger c_\uparrow^\dagger + c_\downarrow c_\uparrow$

chargecharge

`chargecharge[a[i], b[j]]` gives the charge-charge (density-density) interaction $n_{a,i}\, n_{b,j}$ between the levels `a[i]` and `b[j]`.
`chargecharge[a[i], b[j]]` is `nc[number[a[i]], number[b[j]]]`, with $n = \sum_\sigma c_\sigma^\dagger c_\sigma$. The result is normal ordered.
The levels are fermion levels given without the creation/annihilation and spin indexes.
The extended Hubbard model contains the intersite repulsion $V \sum_{\langle ij\rangle} n_i n_j$, written as `V chargecharge[c[i], c[j]]` for each bond.
For the same level, `chargecharge[c[i], c[i]]` is $n_i^2 = n_i + 2n_{i\uparrow}n_{i\downarrow}$, which includes the self-interaction term n_i. The number of electron pairs is $n_i(n_i-1)/2 = n_{i\uparrow}n_{i\downarrow}$ (`hubbard`).

`chargecharge` conserves the charge and the spin.
`chargecharge` is not `Listable`. Bare symbols are not accepted as levels.

Example.

```
snegfermionoperators[c, d];
chargecharge[c[], d[]]
```

current

`current[a[i], b[j]]` gives the particle-current operator $\sum_\sigma(-i\,a^\dagger_{i\sigma}b_{j\sigma} + i\,b^\dagger_{j\sigma}a_{i\sigma})$ associated with `hop[a[i], b[j]]`.
`current[a[i], b[j], σ]` gives the current $-i\,a^\dagger_{i\sigma}b_{j\sigma} + i\,b^\dagger_{j\sigma}a_{i\sigma}$ of the spin projection *σ* only.
`current[a[i], b[j]]` is $i[\mathrm{hop}(a_i, b_j), n_a]$ with $n_a =$ `number[a[i]]`: it is the rate of change $\dot{n}_a$ ($\hbar = 1$) generated by the hopping term +`hop[a[i], b[j]]`, i.e. the particle current flowing from `b[j]` to `a[i]` for the hopping amplitude +1.
For the hopping term $-t$ `hop[a[i], b[j]]` with real t, the particle current from `b[j]` to `a[i]` is $J_{b\to a} = -t$ `current[a[i], b[j]]`. The electric current is obtained by multiplying with the charge of the carriers, $-e$ for electrons.
`current[a[i], b[j]]` is `currentphi[a[i], b[j], 0]`, which equals `hopphi[a[i], b[j], -Pi/2]`; for hopping terms with a phase, see `currentphi`.
`current[b[j], a[i]]` is `-current[a[i], b[j]]`.
The form without *σ* requires spin-1/2 fermions. The levels are given without the creation/annihilation and spin indexes. `current` is not `Listable`.

Example. Current between two levels:

```
snegfermionoperators[a, b];
current[a[], b[]]
```

currentphi

`currentphi[a[i], b[j], φ]` gives the particle-current operator $\sum_\sigma(-ie^{i\phi}a^\dagger_{i\sigma}b_{j\sigma} + ie^{-i\phi}b^\dagger_{j\sigma}a_{i\sigma})$ associated with `hopphi[a[i], b[j], φ]`.
`currentphi[a[i], b[j], σ, φ]` gives the current of the spin projection *σ* only.
`currentphi[a[i], b[j], φ]` is $i[\mathrm{hopphi}(a_i, b_j, \phi), n_a]$ with $n_a =$ `number[a[i]]`, the particle current from `b[j]` to `a[i]` for the tunneling term +`hopphi[a[i], b[j], φ]`. It equals `hopphi[a[i], b[j], φ - Pi/2]` and $-\partial\,\mathrm{hopphi}/\partial\phi$.
For the tunneling term v `hopphi[a[i], b[j], φ]`, the current from `b[j]` to `a[i]` is v `currentphi[a[i], b[j], φ]`.
`currentphi[a[i], b[j], 0]` is `current[a[i], b[j]]`.
Josephson current: if the superconducting phase difference enters the Hamiltonian only through tunneling phases (see `hopphi`), the supercurrent operator follows from

$\partial H/\partial\phi$, and the equilibrium current is $I = (2e/\hbar)\,\partial F/\partial\phi$, where ϕ is the phase difference. For phases $\pm\phi/4$ on the two tunneling terms, $\partial H/\partial\phi$ is a combination of `currentphi` operators with the factors $\mp 1/4$.
The result contains the phase factors in the form `Exp[I (φ - Pi/2)]`; apply `ExpToTrig` or `Simplify` to simplify them.
The form without *σ* requires spin-1/2 fermions. `currentphi` is not `Listable`.

Example. Current operator for tunneling with a phase:

```
snegfermionoperators[a, b];
snegrealconstants[phi];
ExpToTrig[currentphi[a[], b[], phi]]
```

genanhop

`genanhop[t, a[i], b[j]]` gives the singlet anomalous hopping operator with the complex amplitude t, $t\,(a^\dagger_{i\uparrow} b^\dagger_{j\downarrow} - a^\dagger_{i\downarrow} b^\dagger_{j\uparrow}) + \text{h.c.}$
`genanhop[t, a[i], b[j], σ]` gives $t\,a^\dagger_{i\sigma} b^\dagger_{j\bar\sigma} + t^*\, b_{j\bar\sigma} a_{i\sigma}$.
The levels are spin-1/2 fermion levels given without the creation/annihilation and spin indexes; $\bar\sigma$ is the spin opposite to σ.
`genanhop[t, a, b]` is `genanhop[t, a, b, UP] - genanhop[t, a, b, DO]`. The Hermitian conjugate is formed with `Conjugate[t]`.
`genanhop[1, a, b]` is `anhop[a, b]`, and `genanhop[t, a, a]` is `2 bcs[t, a]`.
`genanhop` is the pairing counterpart of `genhop`: a general spin-singlet quadratic coupling between two levels is `genhop[v, a, b] + genanhop[w, a, b]`, as in the hybridization of an impurity with a superconductor in the basis of NRG Ljubljana with spin-SU(2) symmetry but without charge conservation.
`genanhop` is `Listable`.

Example.

```
snegfermionoperators[a, b];
snegcomplexconstants[z];
genanhop[z, a[], b[]]
```

genhop

`genhop[t, a[i], b[j]]` gives the hopping operator $\sum_\sigma (t\,a^\dagger_{i\sigma} b_{j\sigma} + t^*\, b^\dagger_{j\sigma} a_{i\sigma})$ with the complex amplitude t.
`genhop[t, a[i], b[j], σ]` gives $t\,a^\dagger_{i\sigma} b_{j\sigma} + t^*\, b^\dagger_{j\sigma} a_{i\sigma}$ for the spin projection *σ* only.
The levels are spin-1/2 fermion levels given without the creation/annihilation and spin indexes, as in `hop`. The form without *σ* requires spin-1/2 fermions; the form with *σ* works for any fermion operators.

t is any scalar. The Hermitian conjugate is formed with `Conjugate[t]`; for constants declared with `snegrealconstants`, `Conjugate[t]` is t, and for undeclared symbols it remains `Conjugate[t]`.

`genhop[t, a[i], b[j]]` creates an electron on `a[i]` with amplitude t: the order of the levels matters, `genhop[t, a, b]` equals `genhop[Conjugate[t], b, a]`.

Special cases: `genhop[1, a, b]` is `hop[a, b]`, `genhop[Exp[I φ], a, b]` is `hopphi[a, b, φ]`, and `genhop[t, a, b]` equals `matrixhop[t IdentityMatrix[2], a, b]`.

In Hamiltonians, a hopping integral t with the usual sign enters as `genhop[-t, a, b]`. Gauge-invariant quantities depend on the phases of the amplitudes only through the sums of phases around closed loops (magnetic flux) and through the phases relative to superconducting order parameters.

The particle current from `b[j]` to `a[i]` generated by `genhop[t, a[i], b[j]]` is $i[H, n_a] =$ `genhop[-I t, a[i], b[j]]`, i.e. $-it\,a^\dagger b + it^* b^\dagger a$.

`genhop` is not `Listable`. Bare symbols and abstract orbitals (pure functions) are not supported as levels; use `hop` or `matrixhop` for abstract orbitals.

Example. Hopping with a complex amplitude:

```
snegfermionoperators[a, b];
snegcomplexconstants[z];
genhop[z, a[], b[]]
```

hamiltonian

`hamiltonian[Hubbard, c[j], n, U, t, ε]` gives the Hamiltonian of the open Hubbard chain of n sites $\sum_{i=1}^{n}(\epsilon\, n_i + U n_{i\uparrow} n_{i\downarrow}) + t\sum_{i=1}^{n-1}\sum_{\sigma}(c^\dagger_{i\sigma} c_{i+1,\sigma} + \text{h.c.})$.

`hamiltonian` is a stub for model Hamiltonians; only the type `Hubbard` is defined. For other types, `hamiltonian` remains unevaluated.

The site operators are `c[i, j]`, $i = 1, \ldots, n$: the site index is inserted as the first index of the level `c[j]`, which may have further indexes, e.g. a channel.

The hopping enters with the sign $+t$: the usual convention $-t\sum$ `hop` corresponds to the argument $-t$.

The chain has open boundary conditions. For periodic chains, other lattices, or longer-range terms, build the Hamiltonian from `hop`, `number`, and `hubbard` directly.

Example. The Hubbard dimer:

```
snegfermionoperators[c];
snegrealconstants[U, t, ε];
hamiltonian[Hubbard, c[], 2, U, t, ε]
```

holehop

holehop[h[i], p[j]] gives the hopping term $\sum_\sigma (c^\dagger_{i\sigma} p_{j\sigma} + \text{h.c.})$ written with the hole operators $h_{i\bar\sigma} = c^\dagger_{i\sigma}$ of the level h[i]: $\sum_\sigma (h_{i\bar\sigma} p_{j\sigma} + p^\dagger_{j\sigma} h^\dagger_{i\bar\sigma})$.
holehop[h[i], p[j], σ] gives the term for the particle spin *σ* only.
The levels are spin-1/2 fermion levels given without the creation/annihilation and spin indexes. The first argument is the level described by hole operators, the second the level described by particle operators.
The particle-hole transformation used is $c^\dagger_{i\sigma} \to h_{i\bar\sigma}$, $c_{i\sigma} \to h^\dagger_{i\bar\sigma}$, without sign factors: creating an electron with spin σ annihilates a hole with spin $\bar\sigma$. With this transformation, hop[c[i], p[j]] becomes holehop[h[i], p[j]].
Since the transformation has no spin-dependent sign, the hole operators do not transform as a spinor under spin rotations, and holehop does not commute with the conventional total spin operators S^x, S^y built from h and p; it commutes with S^z. (The transformation $c^\dagger_\uparrow \to h_\downarrow$, $c^\dagger_\downarrow \to -h_\uparrow$ would preserve the SU(2) structure.)
holehop[h[i], p[j]] equals anomaloushop[p[j], h[i]].
holehop is Listable.

Example.

```
snegfermionoperators[h, f];
holehop[h[], f[]]
```

hop

hop[a[i], b[j]] gives the hopping operator $\sum_\sigma (a^\dagger_{i\sigma} b_{j\sigma} + b^\dagger_{j\sigma} a_{i\sigma})$ between the levels a[i] and b[j].
hop[a[i], b[j], σ] gives the hopping operator $a^\dagger_{i\sigma} b_{j\sigma} + b^\dagger_{j\sigma} a_{i\sigma}$ for the spin projection *σ* only.
hop[fn1, fn2] gives the hopping between abstract orbitals given as pure functions.
The levels are specified without the creation/annihilation and spin indexes: a[i] stands for a[CR, i, σ] and a[AN, i, σ]. The two levels may belong to the same or to different operator families, e.g. hop[d[], c[0]] for the hybridization of an impurity level with a conduction-band site.
hop is the bare hopping with amplitude +1. The tight-binding Hamiltonian with hopping integral t is $H = -t \sum_{\langle ij \rangle} \text{hop}(c_i, c_j)$; with this sign the band $\epsilon_k = -2t \cos k$ of a chain has its minimum at $k = 0$ for $t > 0$.
Each bond is entered once: $\sum_{\langle ij \rangle}$ runs over unordered pairs, since hop already contains the Hermitian conjugate.
For spin-1/2 fermions the sum runs over $\sigma \in \{\uparrow, \downarrow\}$, for fermions with spin s over $\sigma = -s, \ldots, s$, and spinless fermions have the spin index 0. Both levels must have the same spin; otherwise hop remains unevaluated.

`hop[a[i], a[i]]` is $2n_i$.
Abstract orbitals: `fn` is a pure function of the type and spin indexes such that `fn[CR, σ]` is the creation operator of the orbital. The annihilation operator is obtained as `conj[fn[CR, σ]]`. In `hop[fn1, fn2]`, one argument may be an ordinary level `a[i]`, which is converted to `a[#1, i, #2] &`; the spin-resolved form `hop[fn1, fn2, σ]` requires two functions.
Complex hopping amplitudes: `genhop[t, a[i], b[j]]` gives $t\,a^\dagger b + t^* b^\dagger a$, and `hopphi[a[i], b[j], φ]` the Peierls-phase form $e^{i\phi}a^\dagger b + e^{-i\phi}b^\dagger a$. Spin-dependent hopping $a^\dagger T b$ + h.c. with a 2×2 matrix T is given by `matrixhop`.
The particle current associated with `hop[a[i], b[j]]` is `current[a[i], b[j]]` $= i[\mathrm{hop}, n_a]$.
`hop` is `Listable`: `hop[{a1, a2}, {b1, b2}]` gives `{hop[a1, b1], hop[a2, b2]}`, and `hop[{a1, a2}, b]` gives `{hop[a1, b], hop[a2, b]}`.
`SnegSimplify` rewrites expanded hopping terms as `hop[...]`.

Example. Hopping between two sites:

In[1]:=
```
snegfermionoperators[c];
hop[c[1], c[2]]
```

Out[2]= $c^\dagger_{1\downarrow}c_{2\downarrow} + c^\dagger_{1\uparrow}c_{2\uparrow} + c^\dagger_{2\downarrow}c_{1\downarrow} + c^\dagger_{2\uparrow}c_{1\uparrow}$

hopphi

`hopphi[a[i], b[j], φ]` gives the hopping operator with the phase *φ*, $\sum_\sigma (e^{i\phi}a^\dagger_{i\sigma}b_{j\sigma} + e^{-i\phi}b^\dagger_{j\sigma}a_{i\sigma})$.
`hopphi[a[i], b[j], σ, φ]` gives $e^{i\phi}a^\dagger_{i\sigma}b_{j\sigma} + e^{-i\phi}b^\dagger_{j\sigma}a_{i\sigma}$ for the spin projection *σ* only.
The levels are spin-1/2 fermion levels given without the creation/annihilation and spin indexes, as in `hop`. The form without *σ* requires spin-1/2 fermions.
φ is the phase acquired by an electron that hops from `b[j]` to `a[i]`. `hopphi[a, b, φ]` equals `hopphi[b, a, -φ]` and `genhop[Exp[I φ], a, b]`; `hopphi[a, b, 0]` is `hop[a, b]`.
Peierls substitution: in a vector potential, the hopping $-t\,c_i^\dagger c_j$ acquires the phase $\phi_{ij} = (e/\hbar)\int_j^i \mathbf{A}\cdot d\mathbf{l}$ (up to the sign convention for the charge), $H = -t\sum_{\langle ij\rangle}$ `hopphi[c[i], c[j], φij]`. The sum of the phases around a closed loop is $2\pi\Phi/\Phi_0$, with Φ the magnetic flux through the loop and $\Phi_0 = h/e$.
Gauge transformation: the phase can be moved from the hopping to the level, $a_\sigma \to e^{i\phi}a_\sigma$ turns `hopphi[a, b, φ]` into `hop[a, b]`. A pairing term on that level then acquires a phase: `bcs[a, Φ]` becomes `bcs[a, Φ - 2φ]`. Hence a superconductor with the phase Φ coupled by `hop` is equivalent to a superconductor

with real pairing coupled by `hopphi` with $\phi = -\Phi/2$; this is the form used for phase-biased junctions in NRG Ljubljana, where the superconducting phases are put on the tunneling terms.
The particle current from `b[j]` to `a[i]` for $H =$ `hopphi[a[i], b[j], φ]` is $i[H, n_a] =$ `currentphi[a[i], b[j], φ]`; equivalently, $\partial H/\partial\phi = -$`currentphi[a[i], b[j], φ]`.
`hopphi` is not `Listable`.

Example. Hopping with a phase:

```
snegfermionoperators[a, b];
snegrealconstants[phi];
hopphi[a[], b[], phi]
```

hubbard

`hubbard[c[i]]` gives the on-site Hubbard interaction $n_{i\uparrow}n_{i\downarrow}$ of the fermion level `c[i]`.
`hubbard[c]` is `hubbard[c[]]`.
The level is specified without the creation/annihilation and spin indexes: `c[i]` stands for the operators `c[CR, i, σ]` and `c[AN, i, σ]`; `c[]` is a level without further indexes.
The result is normal ordered: $n_\uparrow n_\downarrow = -c_\downarrow^\dagger c_\uparrow^\dagger c_\downarrow c_\uparrow$.
For fermions with spin $s \neq 1/2$, declared with `snegfermionoperators[{c, s}]`, `hubbard` gives $\sum_{\sigma>\sigma'} n_\sigma n_{\sigma'}$, the number of pairs of electrons in the level.
The Hubbard model is $H = -t\sum_{\langle ij\rangle\sigma}(c_{i\sigma}^\dagger c_{j\sigma} + \text{h.c.}) + U\sum_i n_{i\uparrow}n_{i\downarrow}$. In the Anderson impurity model, $Un_\uparrow n_\downarrow$ is the interaction on the impurity level.
Related forms of the interaction are $\frac{U}{2}(n-1)^2 = Un_\uparrow n_\downarrow - \frac{U}{2}n + \frac{U}{2}$, see `isozsq`, and $\frac{U}{2}n(n-1)$, see `chargecharge`.
`hubbard` is `Listable`.

Example. The Hubbard interaction of a level `d[]`:

In[1]:=
```
snegfermionoperators[d];
hubbard[d[]]
```

Out[2]= $-d_\downarrow^\dagger d_\uparrow^\dagger d_\downarrow d_\uparrow$

isozsq

`isozsq[c[i]]` gives $(n_i - 1)^2$ for the fermion level `c[i]`, four times the square of the z component of the isospin, the particle-hole symmetric form of the on-site interaction.

With $n = n_\uparrow + n_\downarrow$, $(n-1)^2 = 1 - n + 2n_\uparrow n_\downarrow$ for spin-1/2 fermions; it is 1 for the empty and the doubly occupied state and 0 for single occupancy. It equals $4(I^z)^2$ with $I^z = (n-1)/2$.
The interaction term $\frac{U}{2}(n-1)^2 = Un_\uparrow n_\downarrow - \frac{U}{2}n + \frac{U}{2}$ is the Hubbard interaction with the chemical potential at the particle-hole symmetric point and a constant shift. A level $\epsilon\, n + Un_\uparrow n_\downarrow$ can be written as $\frac{U}{2}(n-1)^2 + (\epsilon + U/2)(n-1) + \text{const}$, so that $\delta = \epsilon + U/2$ measures the deviation from particle-hole symmetry.
For fermions with spin s, `isozsq` is $(n-1)^2$ with the corresponding number operator.
`isozsq` is `Listable`.

Example.

```
snegfermionoperators[d];
isozsq[d[]]
```

matrixhop

`matrixhop[T, a[i], b[j]]` gives the spin-dependent hopping operator $\sum_{\sigma\sigma'} a^\dagger_{i\sigma} T_{\sigma\sigma'} b_{j\sigma'}$ + h.c. with the 2×2 matrix T.
In matrix notation, `matrixhop[T, a[i], b[j]]` is $a_i^\dagger T\, b_j + b_j^\dagger T^\dagger a_i$ with the spinors $a_i = (a_{i\uparrow}, a_{i\downarrow})^T$ and $b_j = (b_{j\uparrow}, b_{j\downarrow})^T$. Rows and columns of T are ordered as `{UP, DO}`.
T is a dense matrix or a `SparseArray`. Its entries can be numbers, declared constants, or undeclared symbols; all entries are treated as complex scalars, and the Hermitian conjugate is formed with `Conjugate`. Constants declared with `snegrealconstants` are real.
Special cases: `IdentityMatrix[2]` gives `hop`, `PauliX` gives `spinfliphop`, `z IdentityMatrix[2]` gives `genhop[z, a[i], b[j]]`, and `PauliZ` gives $\text{hop}_\uparrow - \text{hop}_\downarrow$.
A spin-orbit coupled bond with hopping t and spin-orbit amplitude α has $T = -t\,\mathbf{1} + i\,\alpha\cdot\sigma$; a unitary $T = -t\,e^{i\theta\,\mathbf{n}\cdot\sigma}$ describes a spin rotation about $\mathbf{n}$ by the angle -2θ along the bond (SU(2) link variable; $e^{-i\alpha\,\mathbf{n}\cdot\sigma/2}$ is the rotation by α).
The endpoints are spin-1/2 fermion levels given without the creation/annihilation and spin indexes, as in `hop`, or abstract orbitals: pure functions `fn` with `fn[CR, σ]` the creation operator of the orbital; the annihilation operator is `conj[fn[CR, σ]]`.
For symbolic matrices, T is decomposed as $T = \sum_{\mu=0}^{3} t_\mu \sigma_\mu$ with $\sigma_0 = \mathbf{1}$, $t_\mu = \text{Tr}(T\sigma_\mu)/2$. The coefficients are simplified with `ExpToTrig` and `Simplify` under the current `$Assumptions`, and the decomposition is used if it is shorter than the matrix entries. Spin rotations given by `MatrixExp` thus give trigonometric coefficients. Numeric matrices and matrices with inexact numbers are used directly.
If T is not a 2×2 matrix, the message `matrixhop::matrix` is issued and `$Failed` is returned; if an endpoint is not a spin-1/2 fermion level or a pure function,

`matrixhop::op` is issued and `$Failed` is returned.
`matrixhop` is not `Listable`, because the matrix is a single argument.

Example. Spin-orbit hopping $i\alpha\, c_1^\dagger \sigma_y c_2$ + h.c., which flips the spin:

```
snegfermionoperators[c];
snegrealconstants[alpha];
matrixhop[I alpha PauliY, c[1], c[2]]
```

nambu

`nambu[c[i]]` gives the Nambu spinor $\{c_{i\uparrow}^\dagger, c_{i\downarrow}\}$ of the fermion level `c[i]`.
`nambu[c[i], n]` gives $\{c_{i\uparrow}^\dagger, (-1)^n c_{i\downarrow}\}$.
`nambu[c]` is `nambu[c[]]`.
The level is a spin-1/2 fermion level given without the creation/annihilation and spin indexes. The default phase index is $n = 0$.
Both components raise the spin projection by 1/2; with respect to the isospin they form a doublet, with $I^z = +1/2$ for $c_\uparrow^\dagger$ and $I^z = -1/2$ for $c_\downarrow$.
The isospin operators are $I^a = \frac{1}{2}\sum_{\alpha\beta} \eta_\alpha \sigma^a_{\alpha\beta} \eta^\dagger_\beta$ with $\eta =$ `nambu[c[i], n]` and the Pauli matrices σ^a; see `isospin`.
The phase $(-1)^n$ is chosen differently on the two sublattices of a bipartite lattice (e.g. $n = i$ for site i of a chain), so that the total isospin commutes with the hopping (charge SU(2) symmetry of the half-filled Hubbard model).
Note that this spinor is formed from $c_\uparrow^\dagger$ and $c_\downarrow$, the Hermitian conjugate of the Nambu spinor $\Psi = (c_\uparrow, c_\downarrow^\dagger)^T$ used in the Bogoliubov-de Gennes formalism.
`nambu` is `Listable`.

Example.

```
snegfermionoperators[c];
nambu[c[]]
```

number

`number[c[i]]` gives the number operator $n_i = \sum_\sigma c_{i\sigma}^\dagger c_{i\sigma}$ of the fermion level `c[i]`.
`number[c[i], σ]` gives the number operator $n_{i\sigma} = c_{i\sigma}^\dagger c_{i\sigma}$ for the spin projection *σ* only.
`number[b[i]]` gives the number operator $b_i^\dagger b_i$ of the boson mode `b[i]`.
`number[c]` is `number[c[]]`.
The level is specified without the creation/annihilation and spin indexes: `c[i]` stands for the operators `c[CR, i, σ]` and `c[AN, i, σ]`. The level may have any number of indexes, `c[]`, `c[i]`, `c[i, α]`, ...
`number[c[i], σ]` is `nc[c[CR, i, σ], c[AN, i, σ]]` for any *σ*, also symbolic.

For spin-1/2 fermions, $\sigma \in \{\uparrow, \downarrow\}$ (`UP`, `DO`). For fermions with spin s, declared with `snegfermionoperators[{c, s}]`, the sum runs over $\sigma = -s, \ldots, s$. Spinless fermions (`snegspinlessfermionoperators`) carry the spin index 0, as in the basis constructors: `number[f[i]]` is `nc[f[CR, i, 0], f[AN, i, 0]]`.
For bosons, `number[b[i]]` is `nc[b[CR, i], b[AN, i]]`; there is no spin index.
The level can also be an abstract orbital, a pure function `fn` with `fn[CR, σ]` the creation operator of the orbital (a linear combination of creation operators). Then `number[fn, σ]` is $a^\dagger_\sigma a_\sigma$ with $a^\dagger_\sigma$ = `fn[CR, σ]` and a_σ = `conj[fn[CR, σ]]`, and `number[fn]` sums over `UP` and `DO`. The orbital should be normalized for this to be a number operator.
The result is in normal order. The eigenvalues of $n_{i\sigma}$ are 0 and 1, those of n_i for a spin-1/2 level are 0, 1, 2.
The total number operator $N = \sum_i n_i$ generates the U(1) charge symmetry: $[N, c^\dagger_{i\sigma}] = c^\dagger_{i\sigma}$. The charge quantum number $Q = N - N_{\text{half}}$ of `qszbasisvc` is measured from half filling.
`number` is `Listable`.

Example. The number operator of a spin-1/2 level, $n = n_\uparrow + n_\downarrow$:

In[1]:=
```
snegfermionoperators[c];
number[c[]]
```

Out[2]= $c^\dagger_\downarrow c_\downarrow + c^\dagger_\uparrow c_\uparrow$

PROJ0

`PROJ0` specifies the projector onto the empty state in `projector[c[i], PROJ0]`.
`PROJ0` is the protected integer constant 1. The constants `PROJ0`, `PROJUP`, `PROJDO`, `PROJ2`, `PROJ1`, `PROJ02` are 1, ..., 6 (the numbering of T. Rejec's operator-chain code).
`projector[c[i], PROJ0]` is `projector0[c[i]]`, $P_0 = (1 - n_{i\uparrow})(1 - n_{i\downarrow})$.

Example.

```
PROJ0
```

PROJ02

`PROJ02` specifies the projector onto the empty and the doubly occupied states in `projector[c[i], PROJ02]`.
`PROJ02` is the protected integer constant 6. The constants `PROJ0`, `PROJUP`, `PROJDO`, `PROJ2`, `PROJ1`, `PROJ02` are 1, ..., 6 (the numbering of T. Rejec's operator-chain code).
`projector[c[i], PROJ02]` is `projector02[c[i]]`, $P_{02} = P_0 + P_2 = (n_i - 1)^2$.

Example.

PROJ02

PROJ1

PROJ1 specifies the projector onto the singly occupied states in projector[c[i], PROJ1].
PROJ1 is the protected integer constant 5. The constants PROJ0, PROJUP, PROJDO, PROJ2, PROJ1, PROJ02 are 1, ..., 6 (the numbering of T. Rejec's operator-chain code).
projector[c[i], PROJ1] is projector1[c[i]], $P_1 = P_\uparrow + P_\downarrow = n_i - 2n_{i\uparrow}n_{i\downarrow}$.

Example.

PROJ1

PROJ2

PROJ2 specifies the projector onto the doubly occupied state in projector[c[i], PROJ2].
PROJ2 is the protected integer constant 4. The constants PROJ0, PROJUP, PROJDO, PROJ2, PROJ1, PROJ02 are 1, ..., 6 (the numbering of T. Rejec's operator-chain code).
projector[c[i], PROJ2] is projector2[c[i]], $P_2 = n_{i\uparrow}n_{i\downarrow}$.

Example.

PROJ2

PROJDO

PROJDO specifies the projector onto the singly occupied state with spin down in projector[c[i], PROJDO].
PROJDO is the protected integer constant 3. The constants PROJ0, PROJUP, PROJDO, PROJ2, PROJ1, PROJ02 are 1, ..., 6 (the numbering of T. Rejec's operator-chain code).
projector[c[i], PROJDO] is projectorDO[c[i]], $P_\downarrow = n_{i\downarrow}(1 - n_{i\uparrow})$.

Example.

PROJDO

projector

projector[c[i], p] gives the projector onto the subspace of the level c[i] specified by p, one of PROJ0, PROJUP, PROJDO, PROJ2, PROJ1, PROJ02.
projector[c, p] is projector[c[], p].
The level is a spin-1/2 fermion level given without the creation/annihilation and spin indexes. The projectors act on the four states $|0\rangle$, $|\uparrow\rangle$, $|\downarrow\rangle$, $|2\rangle$ of the level, and as the identity on the other levels.

The projectors, written with $n_\sigma = c_\sigma^\dagger c_\sigma$:
`PROJ0 projector0` $P_0 = (1-n_\uparrow)(1-n_\downarrow)$, empty
`PROJUP projectorUP` $P_\uparrow = n_\uparrow(1-n_\downarrow)$, one spin-up electron
`PROJDO projectorDO` $P_\downarrow = n_\downarrow(1-n_\uparrow)$, one spin-down electron
`PROJ2 projector2` $P_2 = n_\uparrow n_\downarrow$, doubly occupied
`PROJ1 projector1` $P_1 = P_\uparrow + P_\downarrow = n - 2n_\uparrow n_\downarrow$, singly occupied
`PROJ02 projector02` $P_{02} = P_0 + P_2 = (n-1)^2$, empty or doubly occupied
`PROJ0`, ..., `PROJ02` are the integers 1, ..., 6; `projector[c[i], k]` with these integers gives the same results. Other values of p leave `projector` unevaluated.
The projectors are diagonal in the occupation numbers, commute with the number and S^z operators of the level, and satisfy $P_a P_b = \delta_{ab} P_a$ and $P_0 + P_\uparrow + P_\downarrow + P_2 = 1$.
Products of projectors on several levels project onto configurations of several sites, e.g. `nc[projector1[c[1]], projector1[c[2]]]` onto the subspace with single occupancy of both sites. They are used to define the low-energy subspace in `schriefferwolff`.
For projectors onto the superpositions $(|0\rangle \pm e^{-i\phi}|2\rangle)/\sqrt{2}$, see `projectorEphi` and `projectorOphi`.
`projector` and the short forms `projector0`, ... are `Listable`, also in the second argument.

Example. The projector onto the singly occupied states:

```
snegfermionoperators[c];
projector[c[], PROJ1]
```

projector0

`projector0[c[i]]` gives the projector $P_0 = (1-n_{i\uparrow})(1-n_{i\downarrow})$ onto the empty state of the level `c[i]`.
`projector0[c]` is `projector0[c[]]`.
The level is a spin-1/2 fermion level given without the creation/annihilation and spin indexes; other levels are not affected.
`projector0[c[i]]` is `projector[c[i], PROJ0]`. See `projector` for the complete set of local projectors and their relations.
In normal-ordered form, $P_0 = 1 - n + n_\uparrow n_\downarrow$.
`projector0` is `Listable`.

Example.

```
snegfermionoperators[c];
projector0[c[]]
```

projector02

`projector02[c[i]]` gives the projector $P_{02} = P_0 + P_2 = (n_i - 1)^2$ onto the empty and the doubly occupied states of the level `c[i]`.
`projector02[c]` is `projector02[c[]]`.
The level is a spin-1/2 fermion level given without the creation/annihilation and spin indexes; other levels are not affected.
`projector02[c[i]]` is `projector[c[i], PROJ02]`. See `projector` for the complete set of local projectors and their relations.
$P_{02} = 1 - P_1$ is the projector onto the states with the isospin 1/2 (charge doublet) of the level; it equals `isozsq[c[i]]`.
`projector02` is `Listable`.

Example.

```
snegfermionoperators[c];
projector02[c[]]
```

projector1

`projector1[c[i]]` gives the projector $P_1 = P_\uparrow + P_\downarrow = n_i - 2n_{i\uparrow}n_{i\downarrow}$ onto the singly occupied states of the level `c[i]`.
`projector1[c]` is `projector1[c[]]`.
The level is a spin-1/2 fermion level given without the creation/annihilation and spin indexes; other levels are not affected.
`projector1[c[i]]` is `projector[c[i], PROJ1]`. See `projector` for the complete set of local projectors and their relations.
P_1 projects onto the local-moment (spin-1/2) states of the level and commutes with all spin components; $\mathbf{S}^2 = \frac{3}{4}P_1$.
Products such as `nc[projector1[c[1]], projector1[c[2]]]` define the low-energy subspace of the half-filled Hubbard model in `schriefferwolff`.
`projector1` is `Listable`.

Example.

In[1]:=
```
snegfermionoperators[c];
projector1[c[]]
```

Out[2]= $c_\downarrow^\dagger c_\downarrow + c_\uparrow^\dagger c_\uparrow + 2c_\downarrow^\dagger c_\uparrow^\dagger c_\downarrow c_\uparrow$

projector2

`projector2[c[i]]` gives the projector $P_2 = n_{i\uparrow}n_{i\downarrow}$ onto the doubly occupied state of the level `c[i]`.

`projector2[c]` is `projector2[c[]]`.
The level is a spin-1/2 fermion level given without the creation/annihilation and spin indexes; other levels are not affected.
`projector2[c[i]]` is `projector[c[i], PROJ2]`. See `projector` for the complete set of local projectors and their relations.
P_2 equals the Hubbard interaction operator `hubbard[c[i]]`.
`projector2` is `Listable`.

Example.

```
snegfermionoperators[c];
projector2[c[]]
```

projectorDO

`projectorDO[c[i]]` gives the projector $P_\downarrow = n_{i\downarrow}(1 - n_{i\uparrow})$ onto the singly occupied state with spin down of the level `c[i]`.
`projectorDO[c]` is `projectorDO[c[]]`.
The level is a spin-1/2 fermion level given without the creation/annihilation and spin indexes; other levels are not affected.
`projectorDO[c[i]]` is `projector[c[i], PROJDO]`. See `projector` for the complete set of local projectors and their relations.
$P_\uparrow - P_\downarrow = 2S^z$ and $P_\uparrow + P_\downarrow = P_1$ (`projector1`).
`projectorDO` is `Listable`.

Example.

```
snegfermionoperators[c];
projectorDO[c[]]
```

projectorE

`projectorE[c[i]]` gives the projector onto the state $(|0\rangle + |2\rangle)/\sqrt{2}$ of the level `c[i]`, with $|2\rangle = c^\dagger_{i\uparrow}c^\dagger_{i\downarrow}|0\rangle$.
`projectorE[c]` is `projectorE[c[]]`.
The operator is $P_E = \frac{1}{2}[P_0 + P_2 + c_\downarrow c_\uparrow + c^\dagger_\uparrow c^\dagger_\downarrow] = [P_0 + P_2 + \mathrm{bcs}(c)]/2$, with P_0 and P_2 the projectors onto the empty and doubly occupied states.
$(|0\rangle + |2\rangle)/\sqrt{2}$ is the ground state of the local pairing term $-\Delta\,\mathrm{bcs}(c)$ with $\Delta > 0$.
`projectorE[c[i]]` is `projectorEphi[c[i], 0]`; `projector0` is the projector onto the orthogonal even state. See `projectorEphi` for details.

Example.

```
snegfermionoperators[c];
projectorE[c[]]
```

projectorEphi

`projectorEphi[c[i], φ]` gives the projector $|E_\phi\rangle\langle E_\phi|$ onto the state $|E_\phi\rangle = (|0\rangle + e^{-i\phi}|2\rangle)/\sqrt{2}$ of the level `c[i]`, with $|2\rangle = c^\dagger_{i\uparrow}c^\dagger_{i\downarrow}|0\rangle$.
`projectorEphi[c, φ]` is `projectorEphi[c[], φ]`.
The level is a spin-1/2 fermion level given without the creation/annihilation and spin indexes. $|0\rangle$ and $|2\rangle$ are the empty and the doubly occupied states of the level; the other levels are not affected.
The operator is

$$P_E(\phi) = \frac{1}{2}[P_0 + P_2 + e^{i\phi}c_\downarrow c_\uparrow + e^{-i\phi}c^\dagger_\uparrow c^\dagger_\downarrow],$$

with the projectors $P_0 = (1-n_\uparrow)(1-n_\downarrow)$ and $P_2 = n_\uparrow n_\downarrow$ (`projector0`, `projector2`).
`projectorOphi[c[i], φ]` is the projector onto the orthogonal state $|O_\phi\rangle = (|0\rangle - e^{-i\phi}|2\rangle)/\sqrt{2}$; $P_E + P_O = P_0 + P_2$ (`projector02`).
In terms of the pairing operator, $P_{E,O}(\phi) = [P_0 + P_2 \pm \mathrm{bcs}(c,-\phi)]/2$. $|E_\phi\rangle$ is therefore the ground state of the local pairing term $-\Delta\,\mathrm{bcs}(c,\Phi)$ $(\Delta > 0)$ for $\phi = -\Phi$, i.e. the BCS state $(|0\rangle + e^{i\Phi}|2\rangle)/\sqrt{2}$ of a superconductor with the phase Φ in the zero-bandwidth (atomic) limit, and $|O_\phi\rangle$ is the excited even state at the energy $+\Delta$.
`projectorEphi[c[i], 0]` is `projectorE[c[i]]`.
`projectorEphi` is not `Listable`.

Example. The projector onto $(|0\rangle + e^{-i\phi}|2\rangle)/\sqrt{2}$:

```
snegfermionoperators[c];
snegrealconstants[phi];
projectorEphi[c[], phi]
```

projector0

`projector0[c[i]]` gives the projector onto the state $(|0\rangle - |2\rangle)/\sqrt{2}$ of the level `c[i]`, with $|2\rangle = c^\dagger_{i\uparrow}c^\dagger_{i\downarrow}|0\rangle$.
`projector0[c]` is `projector0[c[]]`.
The operator is $P_O = \frac{1}{2}[P_0 + P_2 - c_\downarrow c_\uparrow - c^\dagger_\uparrow c^\dagger_\downarrow] = [P_0 + P_2 - \mathrm{bcs}(c)]/2$.
`projector0[c[i]]` is `projectorOphi[c[i], 0]`, and $P_E + P_O = P_0 + P_2$ with $P_E =$ `projectorE[c[i]]`. See `projectorEphi` for details.

Example.

```
snegfermionoperators[c];
projector0[c[]]
```

projectorOphi

`projectorOphi[c[i], φ]` gives the projector $|O_\phi\rangle\langle O_\phi|$ onto the state $|O_\phi\rangle = (|0\rangle - e^{-i\phi}|2\rangle)/\sqrt{2}$ of the level `c[i]`, with $|2\rangle = c^\dagger_{i\uparrow}c^\dagger_{i\downarrow}|0\rangle$.
`projectorOphi[c, φ]` is `projectorOphi[c[], φ]`.
The operator is $P_O(\phi) = \frac{1}{2}[P_0 + P_2 - e^{i\phi}c_\downarrow c_\uparrow - e^{-i\phi}c^\dagger_\uparrow c^\dagger_\downarrow] = [P_0 + P_2 - \mathrm{bcs}(c, -\phi)]/2$.
It is the complement of `projectorEphi` in the even subspace of the level: $P_E(\phi) + P_O(\phi) = P_0 + P_2$. $|O_{-\Phi}\rangle$ is the excited even state of the local pairing term $-\Delta\,\mathrm{bcs}(c, \Phi)$, at the energy $+\Delta$.
`projectorOphi[c[i], 0]` is `projectorO[c[i]]`. See `projectorEphi` for details and applications.

Example.

```
snegfermionoperators[c];
snegrealconstants[phi];
projectorOphi[c[], phi]
```

projectorUP

`projectorUP[c[i]]` gives the projector $P_\uparrow = n_{i\uparrow}(1 - n_{i\downarrow})$ onto the singly occupied state with spin up of the level `c[i]`.
`projectorUP[c]` is `projectorUP[c[]]`.
The level is a spin-1/2 fermion level given without the creation/annihilation and spin indexes; other levels are not affected.
`projectorUP[c[i]]` is `projector[c[i], PROJUP]`. See `projector` for the complete set of local projectors and their relations.
$P_\uparrow - P_\downarrow = 2S^z$ and $P_\uparrow + P_\downarrow = P_1$ (`projector1`).
`projectorUP` is `Listable`.

Example.

```
snegfermionoperators[c];
projectorUP[c[]]
```

PROJUP

`PROJUP` specifies the projector onto the singly occupied state with spin up in `projector[c[i], PROJUP]`.
`PROJUP` is the protected integer constant 2. The constants `PROJ0`, `PROJUP`, `PROJDO`, `PROJ2`, `PROJ1`, `PROJ02` are 1, ..., 6 (the numbering of T. Rejec's operator-chain code).
`projector[c[i], PROJUP]` is `projectorUP[c[i]]`, $P_\uparrow = n_{i\uparrow}(1 - n_{i\downarrow})$.

Example.

```
PROJUP
```

spinfliphop

`spinfliphop[a[i], b[j]]` gives the spin-flip hopping operator $\sum_\sigma (a^\dagger_{i\sigma} b_{j\bar\sigma} + b^\dagger_{j\sigma} a_{i\bar\sigma})$, where $\bar\sigma$ is the spin opposite to σ.
The levels are spin-1/2 fermion levels given without the creation/annihilation and spin indexes.
In matrix notation, `spinfliphop[a, b]` is $a^\dagger \sigma_x b + b^\dagger \sigma_x a$ with the Pauli matrix σ_x; it equals `matrixhop[PauliX, a, b]`. Other spin-dependent hopping matrices, e.g. σ_y for Rashba-type spin-orbit coupling, are given by `matrixhop`.
The spin-flip hopping breaks the spin SU(2) symmetry. It conserves the charge and commutes with the total S^x of the two levels.
`spinfliphop[a[i], a[i]]` is $4S^x_i$.
`spinfliphop` is `Listable`. For a spin-flip hopping with a phase, see `spinfliphopphi`.

Example.

```
snegfermionoperators[c, d];
spinfliphop[c[], d[]]
```

spinfliphopphi

`spinfliphopphi[a[i], b[j], φ]` gives the spin-flip hopping operator with the phase *φ*, $\sum_\sigma (e^{i\phi} a^\dagger_{i\sigma} b_{j\bar\sigma} + e^{-i\phi} b^\dagger_{j\sigma} a_{i\bar\sigma})$.
The levels are spin-1/2 fermion levels given without the creation/annihilation and spin indexes; $\bar\sigma$ is the spin opposite to σ.
It equals `matrixhop[Exp[I φ] PauliX, a[i], b[j]]`, and `spinfliphopphi[a, b, 0]` is `spinfliphop[a, b]`. The phase has the same meaning as in `hopphi`.
`spinfliphopphi[a, b, Pi/2]` is the imaginary spin-flip hopping $i \sum_\sigma (a^\dagger_\sigma b_{\bar\sigma} - b^\dagger_\sigma a_{\bar\sigma})$.
`spinfliphopphi` is `Listable`.

Example.

```
snegfermionoperators[c, d];
snegrealconstants[phi];
spinfliphopphi[c[], d[], phi]
```

spinspin

`spinspin[a[i], b[j]]` gives the scalar product $\mathbf{S}_{a,i} \cdot \mathbf{S}_{b,j}$ of the spin operators of the fermion levels `a[i]` and `b[j]`.
`spinspin[S[i], S[j]]` gives $\mathbf{S}_i \cdot \mathbf{S}_j = \sum_a S^a_i S^a_j$ for abstract spin operators declared with `snegspinoperators`.
`spinspin[a, b]` is `spinspin[a[], b[]]`.

For fermion levels, $\mathbf{S}_a \cdot \mathbf{S}_b = S_a^x S_b^x + S_a^y S_b^y + S_a^z S_b^z$ with the spin operators of `spinxyz`. For spin-1/2 fermions,

$$\mathbf{S}_a \cdot \mathbf{S}_b = S_a^z S_b^z + \frac{1}{2}(S_a^+ S_b^- + S_a^- S_b^+) = \frac{1}{4} \sum_{\sigma\sigma'\tau\tau'} \sigma_{\sigma\sigma'} \cdot \sigma_{\tau\tau'}\, c_{a\sigma}^\dagger c_{a\sigma'} c_{b\tau}^\dagger c_{b\tau'}.$$

The result is normal ordered.

The longitudinal and transverse parts are given by `spinspinz` and `spinspinxy`, with `spinspin[a, b] == spinspinz[a, b] + spinspinxy[a, b]`.

The levels can have different spins, e.g. a spin-1 and a spin-1/2 fermion level.

For abstract spin operators `S`, the result is `nc[S[i, 1], S[j, 1]] + nc[S[i, 2], S[j, 2]] + nc[S[i, 3], S[j, 3]]` in Cartesian components (component indexes 1, 2, 3 for x, y, z). For $i = j$ and a declared spin size s this reduces to $s(s+1)$ (for a symbolic s after `Expand`). Use `spincomponents` to rewrite it in terms of S^z and $S^\pm$. The two arguments can be different spin symbols.

A level can also be given as a function `f` with `f[t, σ]` the operator of type t (`CR` or `AN`) and spin σ, e.g. `(c[#1, 1, #2] + c[#1, 2, #2])/Sqrt[2] &`; see `spinxyz`.

`spinspin` of a fermion level and an abstract spin is not defined and remains unevaluated. Write the Kondo exchange $J\,\mathbf{S}\cdot\mathbf{s}$ with the components, e.g. `Sum[nc[S[a], spinxyz[c[]][[a]]], {a, 3}]`.

In the Heisenberg model $H = J\sum_{\langle ij\rangle} \mathbf{S}_i \cdot \mathbf{S}_j$, $J > 0$ is antiferromagnetic. For two spins 1/2, $\mathbf{S}_1 \cdot \mathbf{S}_2 = -3/4$ in the singlet and $1/4$ in the triplet.

Example. Exchange interaction between two fermion levels:

```
snegfermionoperators[c];
spinspin[c[1], c[2]]
```

spinspinsymmetric

`spinspinsymmetric[{a, b, ...}]` gives the average $\frac{1}{N}\sum_{\text{pairs}} \mathbf{S}_a \cdot \mathbf{S}_b$ of the spin-spin products over all N pairs of the fermion levels a, b, ...

For n levels, $N = n(n-1)/2$. The result is invariant under permutations of the levels.

With the total spin $\mathbf{S} = \sum_a \mathbf{S}_a$, $\sum_{\text{pairs}} \mathbf{S}_a \cdot \mathbf{S}_b = \frac{1}{2}(\mathbf{S}^2 - \sum_a \mathbf{S}_a^2)$.

Levels are given as `c[i]`; the result is simplified with `Simplify`.

Example.

```
snegfermionoperators[c];
Expand[spinspinsymmetric[{c[1], c[2], c[3]}] - (spinspin[c[1], c[2]] +
    spinspin[c[1], c[3]] + spinspin[c[2], c[3]])/3]
```

twohop

`twohop[a[i], b[j]]` gives the two-electron (pair) hopping operator $a^\dagger_{i\downarrow}a^\dagger_{i\uparrow}b_{j\downarrow}b_{j\uparrow} + b^\dagger_{j\downarrow}b^\dagger_{j\uparrow}a_{i\downarrow}a_{i\uparrow}$.
The levels are spin-1/2 fermion levels given without the creation/annihilation and spin indexes.
With the pair operators $B^\dagger_i = a^\dagger_{i\uparrow}a^\dagger_{i\downarrow}$ (the doubly occupied state $|2\rangle = B^\dagger|0\rangle$, as in `bcs`), `twohop[a, b]` is $-(B^\dagger_a B_b + B^\dagger_b B_a)$. The matrix element between the doubly occupied states $|2_a 0_b\rangle$ and $|0_a 2_b\rangle$ is -1 in this convention.
In terms of the isospin operators, `twohop[a, b]` is $-2(I^x_a I^x_b + I^y_a I^y_b)$, the transverse part of an isospin exchange (with the isospin phases $n = 0$ for both levels).
The pair hopping conserves the total charge and the total spin. It arises, e.g., in second order in the hopping between negative-U centers and in multi-orbital Kanamori Hamiltonians.
`twohop` is `Listable`. `SnegSimplify` recognizes the pair-hopping operator.

Example.

```
snegfermionoperators[a, b];
twohop[a[], b[]]
```

15.4 Spin and Isospin

abrikosov

`abrikosov[expr, S -> f]` replaces the spin operators `S` in *expr* by the Abrikosov pseudofermion representation with the fermions `f`.
The representation is $\mathbf{S}_j = \sum_{mm'} f^\dagger_{jm}\mathbf{S}_{mm'}f_{jm'}$, i.e. `S[j, t]` is replaced by the corresponding component of `spinxyz[f[j]]` (with $S^\pm = S^x \pm iS^y$). For spin 1/2, $S^+_j = f^\dagger_{j\uparrow}f_{j\downarrow}$, $S^z_j = \frac{1}{2}(n_{j\uparrow} - n_{j\downarrow})$.
The fermion must have the same spin as the spin operators: `snegfermionoperators[{f, s}]` for `snegspinoperators[{S, s}]` (spin 1/2: `snegfermionoperators[f]`). Otherwise the message `abrikosov::spin` is issued and `$Failed` is returned.
The representation preserves the algebra, but is faithful only in the subspace with one fermion per site, $\sum_m n_{jm} = 1$; the constraint is not imposed. For spin 1/2, the projector onto this subspace is `projector1[f[j]]`.
The result is expanded.

Example.

```
snegspinoperators[{S, 1/2}];
snegfermionoperators[f];
abrikosov[{S[1, 3], S[1, 4], S[1, 5]}, S -> f]
```

fermionstox

`fermionstox[expr, c -> X]` replaces the spin-1/2 fermion operators `c[CR|AN, j..., σ]` in *expr* by the X-operators `X[j..., a, b]` of the Hubbard atom.
The replacement is

$$c_\uparrow^\dagger = X^{\uparrow 0} + X^{2\downarrow}, \quad c_\downarrow^\dagger = X^{\downarrow 0} - X^{2\uparrow}, \quad c_\uparrow = X^{0\uparrow} + X^{\downarrow 2}, \quad c_\downarrow = X^{0\downarrow} - X^{\uparrow 2},$$

with $|2\rangle = c_\uparrow^\dagger c_\downarrow^\dagger |0\rangle$ (see `xoperator`). The products are then reduced with the X-operator algebra.
`c` must be a spin-1/2 fermion operator (message `fermionstox::spin`), and `X` must be declared with `snegxoperators[X, "HubbardAtom"]` (message `xtofermions::states`); otherwise `$Failed` is returned.
`fermionstox` is the inverse of `xtofermions`. A typical use is to express effective Hamiltonians from `schriefferwolff`, which contain projectors onto local occupancies, in terms of X-operators, and then in terms of spins with `xtospin`.

Example.

```
snegfermionoperators[c];
snegxoperators[X, "HubbardAtom"];
{fermionstox[c[CR, 1, UP], c -> X], fermionstox[c[AN, 1, DO], c -> X]}
```

holsteinprimakoff

`holsteinprimakoff[expr, S -> b, n]` applies the Holstein-Primakoff transformation to the spin operators `S` in *expr* and keeps the terms with at most n boson operators.
The transformation is

$$S_j^z = s - b_j^\dagger b_j, \qquad S_j^+ = \sqrt{2s - b_j^\dagger b_j}\, b_j, \qquad S_j^- = b_j^\dagger \sqrt{2s - b_j^\dagger b_j},$$

with the fully polarized state $S^z = s$ as the boson vacuum. The boson of the site `S[j...]` is `b[j...]`, i.e. `b[CR, j...]` and `b[AN, j...]`.
`S` must be declared with a spin size, `snegspinoperators[{S, s}]`, numeric or symbolic; otherwise the message `holsteinprimakoff::size` is issued. `b` must be a boson operator (`snegbosonoperators`), otherwise `holsteinprimakoff::op`. In both cases `$Failed` is returned.
The expression is first rewritten in terms of S^z and $S^\pm$ (`spincomponents`). Each normal-ordered product $(S_j^z)^m (S_j^\pm)^a$ on one site is then expanded exactly in normal-ordered boson monomials, using the Newton series $h(b^\dagger b) = \sum_m \frac{\Delta^m h(0)}{m!} (b^\dagger)^m b^m$ with the forward difference Δ. The coefficients are exact in s, e.g.

$$S^+ = \sqrt{2s}\, b + (\sqrt{2s-1} - \sqrt{2s})\, b^\dagger b\, b + \ldots .$$

This differs from the $1/s$ expansion $\sqrt{2s}\,(1 - b^\dagger b/(4s))\,b$ of the square root, which is recovered for large s.
Terms with more than n boson operators are dropped, also inside sums `sum[...]`.
For linear spin-wave theory, $n = 2$. The commutator of two images truncated at n is exact up to $n - 2$ boson operators.
For an antiferromagnet, rotate the spins of one sublattice with `rotatespin` first, so that the classical ground state is ferromagnetic in the local frames. The resulting quadratic boson Hamiltonian contains anomalous terms $b^\dagger b^\dagger$ and is diagonalized with `bdghamiltonian` and `bogoliubov`.
`holsteinprimakoff` does not thread over lists; map it over the elements (see Possible Issues).
Translation-invariant Hamiltonians written as sums over a lattice index (`sneglatticeindexes`) can be transformed to momentum space with `fourier`.

Example.

```
snegspinoperators[{S, s}];
snegbosonoperators[b];
holsteinprimakoff[#, S -> b, 3] & /@ {S[1, 3], S[1, 4], S[1, 5]}
```

invertspin

`invertspin[expr]` exchanges the spin indexes `UP` and `DO` of all fermion operators in *expr*.
Every spin-1/2 fermion operator $c_{i\sigma}$ is replaced by $c_{i\bar{\sigma}}$, without a sign: this is the spin-flip transformation $\sigma \to -\sigma$, under which $S^z \to -S^z$ and $S^\pm \to S^\mp$. (Time reversal, $c_{i\uparrow} \to c_{i\downarrow}$, $c_{i\downarrow} \to -c_{i\uparrow}$, has an additional sign.)
The spin index must be the last index of the operator. A symbolic spin index σ becomes $1 - \sigma$ (`UP` and `DO` are 1 and 0).
`invertspin` is linear and acts on all factors of products; it is `Listable`.
Only spin-1/2 fermion operators are supported. Other operators are returned inside `invertspin[...]`.

Example.

```
snegfermionoperators[c];
invertspin[nc[c[CR, 1, UP], c[AN, 2, DO]]]
```

isospin

`isospin[c[i]]` gives the list $\{I_i^x, I_i^y, I_i^z\}$ of the isospin (charge pseudospin) operators of the spin-1/2 fermion level `c[i]`.
`isospin[c[i], n]` gives the isospin operators with the phase factor $(-1)^n$.
With the Nambu spinor $\Psi_i = (c_{i\uparrow},\ (-1)^n c_{i\downarrow}^\dagger)^T$, the isospin operators are

$I_i^a = \frac{1}{2}\Psi_i^\dagger \sigma^a \Psi_i, \qquad a = x, y, z,$

i.e. the spin operators of the Nambu spinor. `nambu[c[i], n]` gives $\Psi_i^\dagger = (c_{i\uparrow}^\dagger, (-1)^n c_{i\downarrow})$.
Explicitly,

$$I^z = \frac{1}{2}(n_\uparrow + n_\downarrow - 1), \quad I^+ = (-1)^n c_\uparrow^\dagger c_\downarrow^\dagger, \quad I^- = (-1)^n c_\downarrow c_\uparrow,$$

with $I^\pm = I^x \pm iI^y$, $I^x = \frac{1}{2}(I^+ + I^-)$, $I^y = \frac{1}{2i}(I^+ - I^-)$. For $n = 0$, $I^x = \frac{1}{2}(c_\uparrow^\dagger c_\downarrow^\dagger + c_\downarrow c_\uparrow)$. The operators obey $[I^a, I^b] = i\varepsilon_{abc} I^c$ and commute with the spin operators of the level. The empty and doubly occupied states form an isospin doublet ($I = 1/2$, $I^z = \mp 1/2$); the singly occupied states are isospin singlets ($I = 0$) and spin doublets. Hence $\mathbf{I}^2 + \mathbf{S}^2 = 3/4$ on a single level.
The phase index n is an integer; only its parity matters. Its default value is 0.
On a bipartite lattice, the total isospin $\mathbf{I} = \sum_i \mathbf{I}_i(n_i)$ is formed with n_i even on one sublattice and odd on the other, so that $I^+ = \sum_i (-1)^{n_i} c_{i\uparrow}^\dagger c_{i\downarrow}^\dagger$ is Yang's η-pairing operator at momentum $(\pi, \pi, \ldots)$. With this choice, $\mathbf{I}$ commutes with the hopping between the two sublattices, and the Hubbard model at half filling in the particle-hole symmetric form $H = -t \sum_{\langle ij\rangle\sigma} c_{i\sigma}^\dagger c_{j\sigma} + U \sum_i (n_{i\uparrow} - \frac{1}{2})(n_{i\downarrow} - \frac{1}{2})$ has the charge SU(2) symmetry. In quantum impurity models the phases alternate along the Wilson chain, $n_i = i$.
`isospin` and `isospinxyz` are synonyms. The individual components are given by `isospinx`, `isospiny`, `isospinz`, `isospinplus`, `isospinminus`; the total isospin of several levels and its square by `manyisospin`.
The level is given without the creation/annihilation and spin indexes, `c[i]` or `c[]`; unlike `isospinx`, ..., `isospin` does not accept the bare symbol `c`.
`isospin` is `Listable` in both arguments: `isospin[{c[1], c[2]}, {0, 1}]` gives the isospin operators of the two levels with alternating phases.

Example. Isospin operators of the level `c[]`:

```
snegfermionoperators[c];
isospin[c[]]
```

isospinminus

`isospinminus[c[i]]` gives the isospin-lowering operator $I_i^- = c_{i\downarrow} c_{i\uparrow}$ of the fermion level `c[i]`, which annihilates a pair.
`isospinminus[c[i], n]` gives $I_i^- = (-1)^n c_{i\downarrow} c_{i\uparrow}$.
`isospinminus[c, n]` is `isospinminus[c[], n]`.
$I^- = I^x - iI^y = (I^+)^\dagger$; see `isospin` for the conventions.
`isospinminus` is `Listable`.

Example.

```
snegfermionoperators[c];
{isospinminus[c[]], isospinminus[c[], 1]}
```

isospinplus

`isospinplus[c[i]]` gives the isospin-raising operator $I_i^+ = c_{i\uparrow}^\dagger c_{i\downarrow}^\dagger$ of the fermion level `c[i]`, which creates a pair.
`isospinplus[c[i], n]` gives $I_i^+ = (-1)^n c_{i\uparrow}^\dagger c_{i\downarrow}^\dagger$.
`isospinplus[c, n]` is `isospinplus[c[], n]`.
$I^+ = I^x + iI^y$; see `isospin` for the conventions. In normal order, $c_\uparrow^\dagger c_\downarrow^\dagger$ is displayed as $-c_\downarrow^\dagger c_\uparrow^\dagger$.
$(I^+)^\dagger = I^-$, given by `isospinminus`.
On a bipartite lattice, $\sum_i$ `isospinplus[c[i], i]` with alternating phases is Yang's η-pairing operator; see `isospin`.
`isospinplus` is `Listable`.

Example.

```
snegfermionoperators[c];
{isospinplus[c[]], isospinplus[c[], 1]}
```

isospinx

`isospinx[c[i]]` gives the x component $I_i^x = \frac{1}{2}(c_{i\uparrow}^\dagger c_{i\downarrow}^\dagger + c_{i\downarrow} c_{i\uparrow})$ of the isospin of the fermion level `c[i]`.
`isospinx[c[i], n]` gives I_i^x with the phase factor $(-1)^n$, i.e. $(-1)^n$ times the above.
`isospinx[c, n]` is `isospinx[c[], n]`.
`isospinx` is the first element of `isospin`; see `isospin` for the conventions.
$I^x = (I^+ + I^-)/2$ is proportional to the local pairing operator; `bcs[c[i]]` gives $c_\uparrow^\dagger c_\downarrow^\dagger + c_\downarrow c_\uparrow = 2I^x$.
`isospinx` is `Listable`.

Example.

```
snegfermionoperators[c];
{isospinx[c[]], isospinx[c[], 1]}
```

isospinxyz

`isospinxyz[c[i]]` gives the list $\{I_i^x, I_i^y, I_i^z\}$ of the isospin operators of the fermion level `c[i]`.
`isospinxyz[c[i], n]` uses the phase factor $(-1)^n$.

`isospinxyz` is a synonym of `isospin`; see `isospin` for the definition $I^a = \frac{1}{2}\Psi^\dagger \sigma^a \Psi$ with $\Psi = (c_\uparrow, (-1)^n c_\downarrow^\dagger)^T$, and for the sign conventions on bipartite lattices.
`isospinxyz` is `Listable`.

Example.

```
snegfermionoperators[c];
isospinxyz[c[1]]
```

isospiny

`isospiny[c[i]]` gives the y component $I_i^y = \frac{1}{2i}(c_{i\uparrow}^\dagger c_{i\downarrow}^\dagger - c_{i\downarrow} c_{i\uparrow})$ of the isospin of the fermion level `c[i]`.
`isospiny[c[i], n]` gives I_i^y with the phase factor $(-1)^n$.
`isospiny[c, n]` is `isospiny[c[], n]`.
`isospiny` is the second element of `isospin`; see `isospin` for the conventions.
$I^y = (I^+ - I^-)/(2i)$.
`isospiny` is `Listable`.

Example.

```
snegfermionoperators[c];
isospiny[c[]]
```

isospinz

`isospinz[c[i]]` gives the z component $I_i^z = \frac{1}{2}(n_i - 1)$ of the isospin of the fermion level `c[i]`.
`isospinz[c[i], n]` is the same; the phase factor $(-1)^n$ does not affect I^z.
`isospinz[c, n]` is `isospinz[c[], n]`.
`isospinz` is the third element of `isospin`; see `isospin` for the conventions.
$2I^z = n - 1 = Q$ is the charge measured from half filling, the quantum number Q used by `qszbasis` and `makematricesbzvc`. `isozsq` gives $(n-1)^2 = 4(I^z)^2$.
`isospinz` is `Listable`.

Example.

```
snegfermionoperators[c];
isospinz[c[1]]
```

jordanwigner

`jordanwigner[expr, S -> f, {j1, j2, ...}]` applies the Jordan-Wigner transformation to the spin-1/2 operators `S` of the sites *j1*, *j2*, ... in *expr*, giving an expression in the spinless fermion operators `f`.

For the sites in the given order $j_1, j_2, \ldots$, with k the position of the site in the list,

$$S_k^+ = f_k^\dagger \prod_{l<k}(1-2n_l), \qquad S_k^- = \prod_{l<k}(1-2n_l) f_k, \qquad S_k^z = n_k - \frac{1}{2},$$

with $n_l = f_l^\dagger f_l$, and $S^x = (S^+ + S^-)/2$, $S^y = (S^+ - S^-)/(2i)$. The string $\prod_{l<k}(1-2n_l) = \prod_{l<k} e^{i\pi n_l}$ makes spins on different sites commute.
The spin-up state $|\uparrow\rangle$ corresponds to an occupied fermion level, the spin-down state to an empty one.
`S` must be a spin declared with `snegspinoperators[{S, 1/2}]`, otherwise the message `jordanwigner::spin` is issued; `f` must be a spinless fermion operator (`snegspinlessfermionoperators`), otherwise `jordanwigner::op`. In both cases `$Failed` is returned.
The fermion of the site `S[j]` is `f[j]`, i.e. `f[CR, j]` and `f[AN, j]` (internally with the spin index 0). Sites with several indexes are given as lists, e.g. `{{1, "a"}, {1, "b"}}`.
Only the operators of the listed sites are transformed; the result is expanded.
Nearest-neighbor terms in the chosen ordering give local fermion terms: the XX chain maps to free fermions with the hopping $J/2$, and the S^zS^z coupling to a density-density interaction. Terms between distant sites in the ordering contain the string operators.

Example.

```
snegspinoperators[{S, 1/2}];
snegspinlessfermionoperators[f];
jordanwigner[{S[1, 4], S[2, 4], S[2, 3]}, S -> f, {1, 2}]
```

levicivita

`levicivita[a, b, c]` is the totally antisymmetric Levi-Civita symbol ε_{abc} with indexes 1, 2, 3 and $\varepsilon_{123} = 1$.
For integer arguments, `levicivita[a, b, c]` is `Signature[{a, b, c}]` if all arguments are in $\{1, 2, 3\}$, and 0 otherwise.
For symbolic arguments, the symbol vanishes if two arguments are equal, and the arguments are sorted into canonical order with the sign of the permutation.
`levicivita` is real (`Conjugate[levicivita[a, b, c]]` is `levicivita[a, b, c]`) and has the attribute `NumericFunction`, so it is treated as a scalar in operator products.
The su(2) algebra is $[S^a, S^b] = i\sum_c \varepsilon_{abc} S^c$; with symbolic spin component indexes (`snegspincomponentindexes`), commutators of spin operators are returned in terms of `levicivita`.

Example.

```
{levicivita[1, 2, 3], levicivita[2, 1, 3], levicivita[1, 1, 3],
    levicivita[1, 2, 4]}
```

manyisospin

`manyisospin[{c1, c2, ...}, {n1, n2, ...}]` gives the list $\{I^x, I^y, I^z, \mathbf{I}^2\}$ of the components of the total isospin $\mathbf{I} = \sum_k \mathbf{I}_k(n_k)$ of the fermion levels *ck* with the phases $(-1)^{n_k}$, and its square.
`manyisospin[{c1, c2, ...}]` uses $n_k = 0$ for all levels.
$\mathbf{I}_k(n_k)$ is `isospin[ck, nk]`; see `isospin` for the definition and the sign conventions.
All results are expanded.
On a bipartite lattice, choose n_k even on one sublattice and odd on the other; then the total isospin commutes with the hopping between the sublattices, see `isospin`.
If the two lists have different lengths, the message `manyisospin::unequallength` is issued and `Null` is returned.

Example. Total isospin of two levels with alternating phases:

```
snegfermionoperators[c];
{ix, iy, iz, ii} = manyisospin[{c[1], c[2]}, {0, 1}];
ix
```

manyspin

`manyspin[{a[i], b[j], ...}]` gives the list $\{S^x, S^y, S^z, \mathbf{S}^2\}$ of the components of the total spin $\mathbf{S} = \mathbf{S}_a + \mathbf{S}_b + \cdots$ of the fermion levels and its square.
The components are the sums of `spinxyz` of the levels; $\mathbf{S}^2 = (S^x)^2 + (S^y)^2 + (S^z)^2$.
All results are expanded.
The levels are given without the creation/annihilation and spin indexes. Levels of different fermion families and spins can be combined.
For abstract spin operators, form the sums directly, e.g. `Sum[S[i, 3], {i, n}]`.

Example. Total spin of two levels:

```
snegfermionoperators[c];
{sx, sy, sz, ss} = manyspin[{c[1], c[2]}];
sz
```

PauliMinus

`PauliMinus` is the Pauli matrix $\sigma_x - i\sigma_y$ = `{{0, 0}, {2, 0}}`, twice the spin-lowering matrix σ^-.
Rows and columns are ordered as `{UP, DO}`. See `PauliX` for the conventions and the use of the Pauli matrices in SNEG.

Note the normalization: `PauliPlus` and `PauliMinus` have the matrix element 2, so that $S^{\pm}$ = `PauliPlus/2`, `PauliMinus/2` are the spin-1/2 raising and lowering operators.

Example.

```
MatrixForm[PauliMinus]
```

PauliPlus

`PauliPlus` is the Pauli matrix $\sigma_x + i\sigma_y$ = `{{0, 2}, {0, 0}}`, twice the spin-raising matrix σ^+.
Rows and columns are ordered as `{UP, DO}`. See `PauliX` for the conventions and the use of the Pauli matrices in SNEG.
Note the normalization: `PauliPlus` and `PauliMinus` have the matrix element 2, so that $S^{\pm}$ = `PauliPlus/2`, `PauliMinus/2` are the spin-1/2 raising and lowering operators.

Example.

```
MatrixForm[PauliPlus]
```

PauliX

`PauliX` is the Pauli matrix σ_x = `{{0, 1}, {1, 0}}`.
The Pauli matrices `PauliX`, `PauliY`, `PauliZ` are 2×2 matrices in the basis $(\uparrow, \downarrow)$, i.e. the rows and columns are ordered as `{UP, DO}`. `PauliPlus` and `PauliMinus` are $\sigma_x \pm i\sigma_y$.
The spin-1/2 operators are $S^a = \sigma_a/2$; `spinmatrixX[1/2]` is `PauliX/2`.
The Pauli matrices are used to define the spin operators of fermion levels, $S^a = \frac{1}{2}\sum_{\sigma\sigma'} c^\dagger_\sigma (\sigma_a)_{\sigma\sigma'} c_{\sigma'}$ (`spinxyz`), the isospin operators, and spin-dependent hopping matrices in `matrixhop`.
`PauliX` is a variable holding a matrix, not a function.

Example.

```
MatrixForm /@ {PauliX, PauliY, PauliZ}
```

PauliY

`PauliY` is the Pauli matrix σ_y = `{{0, -I}, {I, 0}}`.
Rows and columns are ordered as `{UP, DO}`. See `PauliX` for the conventions and the use of the Pauli matrices in SNEG.

Example.

In[1]:=
```
MatrixForm[PauliY]
```

Out[1]=
$$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

PauliZ

`PauliZ` is the Pauli matrix σ_z = `{{1, 0}, {0, -1}}`.
Rows and columns are ordered as `{UP, DO}`. See `PauliX` for the conventions and the use of the Pauli matrices in SNEG.

Example.

In[1]:=
```
MatrixForm[PauliZ]
```

Out[1]=
$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

rotatespin

`rotatespin[expr, S, R]` replaces the spin vectors $\mathbf{S}_j$ in *expr* by $R\,\mathbf{S}_j$ for all sites j, with a 3×3 matrix R.
`rotatespin[expr, S, f]` uses the matrix `f[j...]` for the site with the indexes j....
Every operator `S[j..., t]` with $t = 1, \ldots, 5$ is replaced by the corresponding component of $R\,(S_j^x, S_j^y, S_j^z)^T$, i.e. $S_j^a \to \sum_b R_{ab} S_j^b$, and $S^{\pm} \to (R\mathbf{S})^x \pm i(R\mathbf{S})^y$. The result is expanded and contains Cartesian components; use `spincomponents` to rewrite it in terms of S^z and $S^{\pm}$.
For a rotation matrix (orthogonal, determinant 1), the substitution is implemented by a unitary rotation of the spins and preserves the su(2) algebra; $\mathbf{S} \cdot \mathbf{S}'$ of two spins rotated by the same matrix is invariant. Other matrices are not rejected and give general linear transformations of the components.
The function form describes site-dependent rotations: sublattice rotations for antiferromagnets, spirals, and local frames of canted structures. The function receives the site indexes of `S[j..., t]` (without the component) as its arguments; for `S[R, "B", t]` it is called as `f[R, "B"]`.
If the matrix is not 3×3, the message `rotatespin::matrix` is issued and `$Failed` is returned.
The Dzyaloshinskii-Moriya interaction $\mathbf{D} \cdot (\mathbf{S}_1 \times \mathbf{S}_2)$ can be written with `levicivita` or with the cross product of the component vectors.

Example. Rotation by π about the x axis:
```
snegspinoperators[S];
rotatespin[{S[1, 3], S[1, 4]}, S, RotationMatrix[Pi, {1, 0, 0}]]
```

schwingerbosons

`schwingerbosons[expr, S -> {a, b}]` replaces the spin operators `S` in *expr* by the Schwinger-boson representation with the bosons `a` and `b`.
The representation is

$$S_j^+ = a_j^\dagger b_j, \qquad S_j^- = b_j^\dagger a_j, \qquad S_j^z = \frac{1}{2}(a_j^\dagger a_j - b_j^\dagger b_j),$$

with $S^x = (S^+ + S^-)/2$, $S^y = (S^+ - S^-)/(2i)$. The bosons of the site `S[j...]` are `a[j...]` and `b[j...]`.
The representation reproduces the su(2) algebra for any boson state; the spin size is fixed by the constraint $a_j^\dagger a_j + b_j^\dagger b_j = 2s$, which is not imposed by `schwingerbosons`. Then $\mathbf{S}_j^2 = s(s+1)$.
The spin can be declared with or without a size. `a` and `b` must be boson operators, otherwise the message `schwingerbosons::op` is issued and `$Failed` is returned.
The result is expanded and normal ordered.

Example.

```
snegspinoperators[S];
snegbosonoperators[a, b];
schwingerbosons[{S[1, 3], S[1, 4], S[1, 5]}, S -> {a, b}]
```

snegspincomponentindexes

`snegspincomponentindexes[a, b, ...]` declares *a*, *b*, ... as symbolic Cartesian spin component indexes with the values 1, 2, 3 (x, y, z).
Spin component indexes are summation indexes (they are also declared with `snegfreeindexes`) and real: `Conjugate[a]` is *a*, so that `conj[S[j, a]]` is `S[j, a]`.
The commutator of spin operators with symbolic components is $[S_i^a, S_j^b] = i\delta_{ij} \sum_c \varepsilon_{abc} S_j^c$, returned as `I sum[levicivita[a, b, c] S[j, c], {c}]` with a new dummy index *c*. Components 4 and 5 ($S^\pm$) are combined with symbolic components through $S^\pm = S^x \pm iS^y$.
A sum over a component index of an expression that does not contain it gives a factor 3.
`spincomponents` expands sums over component indexes into explicit components.
The declared indexes are listed in `listspincomponentindexes` and recognized by `spincomponentindexQ`.

Example.

```
snegspinoperators[S];
snegspincomponentindexes[a, b];
komutator[S[1, a], S[1, b]]
```

spinall

`spinall[c[i]]` gives the list $\{S_i^x, S_i^y, S_i^z, \mathbf{S}_i^2\}$ for the fermion level `c[i]`.
The first three elements are `spinxyz[c[i]]`, the last is `spinss[c[i]]`; see `spinxyz` for the conventions.
`manyspin` gives the same list for the total spin of several levels.
`spinall` is `Listable`.

Example.

```
snegfermionoperators[c];
spinall[c[]]
```

spinclustermatrices

`spinclustermatrices[H, {S[1], S[2], ...}]` gives the matrices of the spin Hamiltonian H in the subspaces of total S^z, as a list of pairs `{{Sz}, matrix}`.
`spinclustermatrices[H, {S -> {j1, j2, ...}}]` specifies the sites by their indexes.
`spinclustermatrices[H, sites, "Blocks" -> False]` gives the matrix in the full product space.
H is a polynomial in spin operators `S[j..., t]` (components $t = 1, \ldots, 5$ for S^x, S^y, S^z, S^+, S^-) of the listed sites, with scalar coefficients. All spins must have a declared numeric size (`snegspinoperators[{S, s}]`); different spin symbols and sizes can be combined.
The basis is the product basis $|m_1, m_2, \ldots, m_N\rangle$ with the sites in the given order, each m_k running from $+s_k$ down to $-s_k$, and the first site varying slowest. The matrices are built from Kronecker products of the spin matrices `spinmatrixX[s]`, ..., so the computation does not use the operator algebra and is fast.
With `"Blocks" -> True` (default), the matrix is split into the subspaces of total $S^z = \sum_k m_k$, in increasing order of S^z; within a subspace, the states are in the order of the product basis.
The site specification is a list of elements `S[j...]` or rules `S -> {j1, j2, ...}`. Since the last index of a spin operator is the component, `S[0]` evaluates to 0; site 0 must therefore be given as `S -> {0, ...}`. Sites with several indexes are given as `S -> {{1, "a"}, {1, "b"}}`.
The following options can be given:
`"Blocks" True` split the matrix into the subspaces of total S^z
`"Sparse" False` return `SparseArray` objects instead of dense matrices
Failure behavior: an operator that does not belong to the listed sites (or a non-spin operator) gives `spinclustermatrices::op`; an operator that does not conserve the total S^z with `"Blocks" -> True` gives `spinclustermatrices::sz`; a spin without a

declared numeric size or an invalid site specification gives a message from the site parser. In all cases `$Failed` is returned.
For spins coupled to fermions, use `spinketbrarules` and `QSZaddspinket` with `makematricesbzvc`.

Example. Heisenberg dimer of two spins 1/2: triplet $J/4$ and singlet $-3J/4$:

```
snegspinoperators[{S, 1/2}];
snegrealconstants[J];
spinclustermatrices[J spinspin[S[1], S[2]], {S[1], S[2]}]
```

spincomponentindexQ

`spincomponentindexQ[a]` gives `True` if a has been declared as a symbolic spin component index with `snegspincomponentindexes`, and `False` otherwise.
The explicit component values 1, ..., 5 are not spin component indexes in this sense.
The declaration sets the upvalue `spincomponentindexQ[a] ^= True`.

Example.

```
snegspincomponentindexes[a];
{spincomponentindexQ[a], spincomponentindexQ[b], spincomponentindexQ[3]}
```

spincomponents

`spincomponents[expr]` rewrites all spin operators in *expr* in terms of S^z and $S^\pm$.
`spincomponents[expr, "XYZ"]` rewrites them in terms of S^x, S^y, S^z.
A spin operator `S[j..., t]` declared with `snegspinoperators` carries the component t as its last index: 1, 2, 3, 4, 5 for S^x, S^y, S^z, S^+, S^- (the constants `SPININDEXx`, ..., `SPININDEXm`).
The default form `"ZPM"` replaces $S^x = (S^+ + S^-)/2$ and $S^y = (S^+ - S^-)/(2i)$; the form `"XYZ"` replaces $S^\pm = S^x \pm iS^y$. The result is expanded, and products are brought to the canonical order by `nc`.
Products of spin operators are ordered automatically, but Cartesian and ladder components are not converted into each other. Expressions that mix them, such as commutators of $\mathbf{S}_1 \cdot \mathbf{S}_2$ (written with S^x, S^y, S^z) with S^+, therefore need `spincomponents` to reach a unique form in which vanishing expressions become 0.
For spins with a declared numeric size $s > 1/2$, same-site products are reduced to the normal form in S^z and $S^\pm$, so the `"XYZ"` form is stable only for $s = 1/2$ and for spins without a declared size.
Sums `sum[..., {a, ...}]` over spin component indexes declared with `snegspincomponentindexes` are expanded into the three terms $a = x, y, z$; other summation indexes are kept.

Any other value of the second argument gives the message `spincomponents::basis` and returns `$Failed`.

Example.

```
snegspinoperators[S];
spincomponents[S[1, 1] + S[1, 2]]
```

spindown

`spindown[v]` applies the total spin-lowering operator S^- of the levels in the state v (in the creation-operator representation) and normalizes the result.
The levels are determined from the operators `c[t, j..., σ]` that appear in v; the total $S^- = \sum_j S^-_{c[j]}$ of these levels (see `spinminus`) is applied. Parts that annihilate the vacuum are dropped (`zeroonvac`), and the result is divided by its norm.
v is a creation-operator string or a linear combination of such strings, optionally followed by `VACUUM`. If $S^- v = 0$, the result is 0.
Starting from the highest-weight state $|S, S_z = S\rangle$ of a multiplet, repeated application gives the states $|S, S_z\rangle$ with the standard phases, since all matrix elements of S^- are positive. `fixspin` applies it the required number of times.
Only spin-1/2 fermions are supported.

Example.

```
snegfermionoperators[c];
spindown[c[CR, 1, UP]]
```

spinmatrixX

`spinmatrixX[s]` gives the $(2s+1) \times (2s+1)$ matrix of the spin component S^x in the spin-s representation.
The basis is $|s, m\rangle$ with $m = s, s-1, \ldots, -s$ (row and column 1 correspond to $m = s$).
The matrices of the spin-s representation are

$$\langle m'|S^z|m\rangle = m\,\delta_{m'm}, \qquad \langle m'|S^\pm|m\rangle = \sqrt{s(s+1) - m(m \pm 1)}\;\delta_{m',m\pm1},$$

with the standard (Condon-Shortley) phase convention: all matrix elements of $S^\pm$ are real and non-negative. $S^x = (S^+ + S^-)/2$ and $S^y = (S^+ - S^-)/(2i)$.
The five functions `spinmatrixX`, `spinmatrixY`, `spinmatrixZ`, `spinmatrixP` (S^+), `spinmatrixM` (S^-) are defined for integer and half-integer s (also $s = 0$); for other arguments they remain unevaluated.
For $s = 1/2$, the matrices are half the Pauli matrices: `spinmatrixX[1/2] == PauliX/2`.

These matrices define the spin-s fermion operators of `spinxyz`, the bra-ket operators `spinketbraX`, ..., the reduction of products of spin operators with a declared size, and the matrices of `spinclustermatrices`.

Example.

```
spinmatrixX[1] // MatrixForm
```

spinminus

`spinminus[c[i]]` gives the spin-lowering operator $S_i^- = S_i^x - iS_i^y = c_{i\downarrow}^\dagger c_{i\uparrow}$ of the fermion level `c[i]`.
`spinminus[c]` is `spinminus[c[]]`.
See `spinxyz` for the definition of the spin operators. For fermions with spin $s \neq 1/2$, $S^- = \sum_m \sqrt{s(s+1) - m(m-1)}\, c_{m-1}^\dagger c_m$.
$(S^-)^\dagger = S^+$, given by `spinplus`.
`spindown` applies the total S^- of all levels in an expression to a state and normalizes the result.
`spinminus` is `Listable`.

Example.

In[1]:=
```
snegfermionoperators[c];
spinminus[c[1]]
```

Out[2]= $c_{1\downarrow}^\dagger c_{1\uparrow}$

spinplus

`spinplus[c[i]]` gives the spin-raising operator $S_i^+ = S_i^x + iS_i^y = c_{i\uparrow}^\dagger c_{i\downarrow}$ of the fermion level `c[i]`.
`spinplus[c]` is `spinplus[c[]]`.
See `spinxyz` for the definition of the spin operators. For fermions with spin $s \neq 1/2$, $S^+ = \sum_m \sqrt{s(s+1) - m(m+1)}\, c_{m+1}^\dagger c_m$.
$(S^+)^\dagger = S^-$, given by `spinminus`.
`spinplus` is `Listable`.

Example.

In[1]:=
```
snegfermionoperators[c];
spinplus[c[1]]
```

Out[2]= $c_{1\uparrow}^\dagger c_{1\downarrow}$

spinspinmp

`spinspinmp[a[i], b[j]]` gives the spin-flip term $S^-_{a,i}S^+_{b,j}$ between the fermion levels `a[i]` and `b[j]`.
`spinspinmp[a, b]` is `nc[spinminus[a], spinplus[b]]`, without a factor 1/2. The transverse exchange is `spinspinxy[a, b] == (spinspinpm[a, b] + spinspinmp[a, b])/2`.
`spinspinpm[a, b]` gives $S^+_a S^-_b$, the Hermitian conjugate.
The levels must be given with brackets, `c[]` or `c[i]`.

Example.

```
snegfermionoperators[c];
spinspinmp[c[1], c[2]]
```

spinspinpm

`spinspinpm[a[i], b[j]]` gives the spin-flip term $S^+_{a,i}S^-_{b,j}$ between the fermion levels `a[i]` and `b[j]`.
`spinspinpm[a, b]` is `nc[spinplus[a], spinminus[b]]`, without a factor 1/2. The transverse exchange is `spinspinxy[a, b] == (spinspinpm[a, b] + spinspinmp[a, b])/2`.
`spinspinmp[a, b]` gives $S^-_a S^+_b$, the Hermitian conjugate.
The levels must be given with brackets, `c[]` or `c[i]`.

Example.

```
snegfermionoperators[c];
spinspinpm[c[1], c[2]]
```

spinspinx

`spinspinx[a[i], b[j]]` gives the term $S^x_{a,i}S^x_{b,j}$ of the exchange interaction between the fermion levels `a[i]` and `b[j]`.
`spinspinx[a, b]` is `Expand[nc[spinx[a], spinx[b]]]`. Together with `spinspiny` and `spinspinz` it describes fully anisotropic (XYZ) exchange
$J_x S^x_a S^x_b + J_y S^y_a S^y_b + J_z S^z_a S^z_b$.
The levels must be given with brackets, `c[]` or `c[i]`.

Example.

```
snegfermionoperators[c];
spinspinx[c[1], c[2]]
```

spinspinxy

`spinspinxy[a[i], b[j]]` gives the transverse part $S^x_{a,i}S^x_{b,j} + S^y_{a,i}S^y_{b,j} = \frac{1}{2}(S^+_{a,i}S^-_{b,j} + S^-_{a,i}S^+_{b,j})$ of the exchange interaction between the fermion levels `a[i]` and `b[j]`.
`spinspinxy[a, b]` is `(spinspinpm[a, b] + spinspinmp[a, b])/2`; see `spinxyz` for the spin operators.
`spinspin[a, b] == spinspinz[a, b] + spinspinxy[a, b]`: the factor 1/2 is included in `spinspinxy`, but not in `spinspinpm` and `spinspinmp`.
The levels must be given with brackets, `c[]` or `c[i]`; abstract spin operators are not supported.
The result is not expanded.

Example.

```
snegfermionoperators[c];
spinspinxy[c[1], c[2]]
```

spinspiny

`spinspiny[a[i], b[j]]` gives the term $S^y_{a,i}S^y_{b,j}$ of the exchange interaction between the fermion levels `a[i]` and `b[j]`.
`spinspiny[a, b]` is `Expand[nc[spiny[a], spiny[b]]]`; see `spinspinx`.
The levels must be given with brackets, `c[]` or `c[i]`.

Example.

```
snegfermionoperators[c];
spinspiny[c[1], c[2]]
```

spinspinz

`spinspinz[a[i], b[j]]` gives the longitudinal part $S^z_{a,i}S^z_{b,j}$ of the exchange interaction between the fermion levels `a[i]` and `b[j]`.
`spinspinz[a, b]` is `nc[spinz[a], spinz[b]]`; see `spinxyz` for the spin operators.
`spinspin[a, b] == spinspinz[a, b] + spinspinxy[a, b]`. The anisotropic (XXZ) exchange is $J_z S^z_a S^z_b + J_\perp(S^x_a S^x_b + S^y_a S^y_b)$.
The levels must be given with brackets, `c[]` or `c[i]`; abstract spin operators are not supported.
The result is not expanded.

Example.

```
snegfermionoperators[c];
spinspinz[c[1], c[2]]
```

spinss

`spinss[c[i]]` gives the square of the spin, $\mathbf{S}_i^2 = (S_i^x)^2 + (S_i^y)^2 + (S_i^z)^2$, of the fermion level `c[i]`.
`spinss[c]` is `spinss[c[]]`.
`spinss[c[i]]` is the fourth element of `spinall[c[i]]`, computed as $\mathbf{S}\cdot\mathbf{S}$ from `spinxyz`.
For spin-1/2 fermions, $\mathbf{S}^2 = \frac{3}{4}(n - 2n_\uparrow n_\downarrow)$: the eigenvalue is $s(s+1) = 3/4$ for single occupancy and 0 for the empty and doubly occupied states.
For fermions with spin $s \neq 1/2$ the result is returned unexpanded; apply `Expand`.
For the total spin of several levels use `manyspin`.
`spinss` is `Listable`.

Example.

```
snegfermionoperators[c];
spinss[c[]]
```

spintensor

`spintensor[k, q, S[j]]` gives the spherical tensor operator T_q^k of rank k and component q built from the spin `S[j]`.
`spintensor[k, q, S -> j]` is the same; this form also allows the site `0`.
The operators are defined by

$$T_k^k = (-S^+/\sqrt{2})^k, \qquad T_{q-1}^k = \frac{[S^-, T_q^k]}{\sqrt{(k+q)(k-q+1)}},$$

so that they satisfy the defining commutation relations of a spherical tensor, $[S^z, T_q^k] = q\,T_q^k$ and $[S^\pm, T_q^k] = \sqrt{k(k+1) - q(q\pm 1)}\,T_{q\pm 1}^k$.
In particular $T_0^1 = S^z$, $T_{\pm 1}^1 = \mp S^\pm/\sqrt{2}$, and $T_0^2 = (3(S^z)^2 - \mathbf{S}^2)/\sqrt{6}$.
The results are expanded polynomials in S^z and $S^\pm$. For spins with a declared size, products are reduced to the normal form, and operators with $k > 2s$ vanish. For spins without a declared size, S^+S^- products remain.
k and q must be integers with $k \geq 0$ and $|q| \leq k$; otherwise the message `spintensor::range` is issued and `$Failed` is returned. The spin must be an operator declared with `snegspinoperators`.
`stevens` gives the Stevens operators, the Hermitian combinations of $T_{\pm q}^k$ used for crystal fields.
Results are cached for each spin symbol and site.

Example. The rank-1 tensor:

```
snegspinoperators[S];
Table[spintensor[1, q, S[1]], {q, -1, 1}]
```

spinx

`spinx[c[i]]` gives the x component $S_i^x = \frac{1}{2}(c_{i\uparrow}^\dagger c_{i\downarrow} + c_{i\downarrow}^\dagger c_{i\uparrow})$ of the spin of the fermion level `c[i]`.
`spinx[c]` is `spinx[c[]]`.
`spinx[c[i]]` is the first element of `spinxyz[c[i]]`; see `spinxyz` for the definition and conventions. For fermions with spin $s \neq 1/2$, $S^x = \sum_{mm'} c_m^\dagger (S^x)_{mm'} c_{m'}$ with `spinmatrixX[s]`.
$S^x = (S^+ + S^-)/2$.
`spinx` is `Listable`.

Example.

```
snegfermionoperators[c];
spinx[c[1]]
```

spinxyz

`spinxyz[c[i]]` gives the list $\{S_i^x, S_i^y, S_i^z\}$ of the spin operators of the fermion level `c[i]`.
`spinxyz[c]` is `spinxyz[c[]]`.
`spinxyz[f]` for a pure function f with `f[t, σ]` the operator of type t (`CR` or `AN`) and spin σ gives the spin operators of the generalized spin-1/2 level described by f.
For a spin-1/2 fermion `c`, the spin operators are the bilinears

$$S_i^a = \frac{1}{2} \sum_{\sigma\sigma'} c_{i\sigma}^\dagger \, \sigma_{\sigma\sigma'}^a \, c_{i\sigma'}, \qquad a = x, y, z,$$

with the Pauli matrices σ^a (`PauliX`, `PauliY`, `PauliZ`) in the basis $(\uparrow, \downarrow)$. Explicitly, $S^z = \frac{1}{2}(n_\uparrow - n_\downarrow)$, $S^x = \frac{1}{2}(c_\uparrow^\dagger c_\downarrow + c_\downarrow^\dagger c_\uparrow)$, $S^y = \frac{1}{2i}(c_\uparrow^\dagger c_\downarrow - c_\downarrow^\dagger c_\uparrow)$.
The level is specified without the creation/annihilation and spin indexes: `c[i]` stands for the operators `c[CR, i, σ]` and `c[AN, i, σ]`.
For fermions with spin $s \neq 1/2$, declared with `snegfermionoperators[{c, s}]`, $S^a = \sum_{mm'} c_m^\dagger (S^a)_{mm'} c_{m'}$ with the spin-s matrices `spinmatrixX[s]`, ... and $m = s, s-1, \ldots, -s$.
The operators obey $[S^a, S^b] = i\varepsilon_{abc} S^c$ and commute with the number operator. They act in the singly occupied subspace as spin-1/2 operators, and annihilate the empty and the doubly occupied states.
The results are normal ordered and not further simplified.
`spinxyz` is `Listable`: `spinxyz[{c[1], c[2]}]` gives a list of two triples.

The individual components are given by `spinx`, `spiny`, `spinz`; `spinplus` and `spinminus` give $S^{\pm} = S^x \pm iS^y$, `spinss` gives $\mathbf{S}^2$, and `spinall` gives $\{S^x, S^y, S^z, \mathbf{S}^2\}$.
The total spin of several levels is given by `manyspin`.
With a function argument, `spinxyz[f]` is `spinxyzgen[f[CR, #] &]`; the function receives the operator type and the spin index, as in `c[#1, 1, #2] &`. This describes, for instance, bonding orbitals $(c_{1\sigma} + c_{2\sigma})/\sqrt{2}$.
Abstract spin operators that are not composed of fermions, e.g. impurity spins in the Kondo model, are declared with `snegspinoperators`.

Example. Spin operators of the level `c[]`:

```
snegfermionoperators[c];
spinxyz[c[]]
```

spinxyzgen

`spinxyzgen[g]` gives the spin-1/2 operators $\{S^x, S^y, S^z\}$ of the generalized level with the creation operators `g[UP]` and `g[DO]`.
`spinxyzgen[g, s]` gives the spin-s operators of the level with the creation operators `g[m]`, $m = s, s-1, \ldots, -s$.
g is a function of the spin index that returns a creation operator or a linear combination of creation operators, e.g. `c[CR, 1, #] &` or `(c[CR, 1, #] + c[CR, 2, #])/Sqrt[2] &`. The annihilation operators are obtained with `conj`.
With $g^\dagger_\sigma$ = `g[σ]`, the result is $S^a = \frac{1}{2}\sum_{\sigma\sigma'} g^\dagger_\sigma \sigma^a_{\sigma\sigma'} g_{\sigma'}$ in the basis $(\uparrow, \downarrow)$, and $S^a = \sum_{mm'} g^\dagger_m (S^a)_{mm'} g_{m'}$ with the spin-s matrices `spinmatrixX[s]`, ... for the second form.
The operators are spin operators only if the g_σ satisfy canonical anticommutation relations, e.g. for normalized orthogonal combinations of levels.
The results are not expanded.
`spinxyz[c[i]]` is `spinxyzgen[c[CR, i, #] &]`, and `spinxyz[f]` for a function of two arguments (operator type and spin) is `spinxyzgen[f[CR, #] &]`.

Example. Spin of the bonding orbital $b^\dagger_\sigma = (c^\dagger_{1\sigma} + c^\dagger_{2\sigma})/\sqrt{2}$:

```
snegfermionoperators[c];
Expand[spinxyzgen[(c[CR, 1, #] + c[CR, 2, #])/Sqrt[2] &][[3]]]
```

spiny

`spiny[c[i]]` gives the y component $S^y_i = \frac{1}{2i}(c^\dagger_{i\uparrow} c_{i\downarrow} - c^\dagger_{i\downarrow} c_{i\uparrow})$ of the spin of the fermion level `c[i]`.
`spiny[c]` is `spiny[c[]]`.

`spiny[c[i]]` is the second element of `spinxyz[c[i]]`; see `spinxyz` for the definition and conventions. For fermions with spin $s \neq 1/2$, $S^y = \sum_{mm'} c^\dagger_m (S^y)_{mm'} c_{m'}$ with `spinmatrixY[s]`.
$S^y = (S^+ - S^-)/(2i)$.
`spiny` is `Listable`.

Example.

```
snegfermionoperators[c];
spiny[c[1]]
```

spinz

`spinz[c[i]]` gives the z component $S^z_i = \frac{1}{2}(n_{i\uparrow} - n_{i\downarrow})$ of the spin of the fermion level `c[i]`.
`spinz[c]` is `spinz[c[]]`.
`spinz[c[i]]` is the third element of `spinxyz[c[i]]`; see `spinxyz` for the definition and conventions. For fermions with spin $s \neq 1/2$, $S^z = \sum_m m\, n_m$.
The Zeeman term of a level in a magnetic field along z is proportional to S^z, e.g. $g\mu_B B S^z$ for electrons.
`spinz` is `Listable`.

Example.

In[1]:=
```
snegfermionoperators[c];
spinz[c[1]]
```

Out[2]=
$$\frac{c^\dagger_{1\uparrow} c_{1\uparrow} - c^\dagger_{1\downarrow} c_{1\downarrow}}{2}$$

stevens

`stevens[k, q, S[j]]` gives the Stevens operator O^q_k of rank k and component q for the spin `S[j]` with a declared size.
`stevens[k, q, S -> j]` is the same.
The Stevens operators are Hermitian polynomials in the spin components that transform like the tesseral harmonics. The normalization is that of the tables of Abragam and Bleaney; for example, with $X = s(s+1)$,

$$O^0_2 = 3(S^z)^2 - X, \quad O^2_2 = (S^x)^2 - (S^y)^2 = \frac{1}{2}[(S^+)^2 + (S^-)^2], \quad O^{-2}_2 = \frac{1}{2i}[(S^+)^2 - (S^-)^2],$$

$$O^1_2 = \frac{1}{4}[S^z(S^+ + S^-) + (S^+ + S^-)S^z], \quad O^0_4 = 35(S^z)^4 - 30X(S^z)^2 + 25(S^z)^2 - 6X + 3X^2, \quad O^4_4 = \frac{1}{2}[(S^+)^4 + (S^-)^4].$$

They are constructed from the spherical tensors T^k_q of `spintensor`: $O^0_k \propto T^k_0$ and, for $q > 0$, $O^{\pm q}_k \propto (-1)^q T^k_q \pm T^k_{-q}$. The normalization fixes the coefficient of $(S^z)^{k-q}(S^+)^q$

to $L/2$ ($L/(2i)$ for $-q$), and that of $(S^z)^k$ to L for $q = 0$, where L is the leading coefficient of the primitive integer polynomial proportional to $d^q P_k(z)/dz^q$, with the Legendre polynomial P_k.
The results are expanded in S^z and $S^\pm$ and reduced to the normal form for the declared spin size, numeric or symbolic. The operators vanish for $k > 2s$.
A crystal-field Hamiltonian is $H = \sum_{kq} B_k^q O_k^q$. The single-ion anisotropy $D(S^z)^2 + E[(S^x)^2 - (S^y)^2]$ is $\frac{D}{3} O_2^0 + E\, O_2^2$ up to a constant.
The spin must have a declared size (`snegspinoperators[{S, s}]`); otherwise the message `stevens::size` is issued and `$Failed` is returned. k and q must be integers with $k \geq 0$ and $|q| \leq k$ (message `spintensor::range`). `stevens[0, 0, S[j]]` is 1.

Example. The rank-2 Stevens operators of a spin 1:

```
snegspinoperators[{T, 1}];
Table[stevens[2, q, T[1]], {q, -2, 2}]
```

xoperator

`xoperator[c[j], a, b]` gives the Hubbard X-operator $X_j^{ab} = |a\rangle\langle b|$ of the Hubbard atom of the spin-1/2 fermion level `c[j]`, expressed with fermion operators.
`xoperator[c, a, b]` is `xoperator[c[], a, b]`.
The states of the Hubbard atom are labeled by the strings `"0"`, `"UP"`, `"DO"`, `"2"`:

$$|0\rangle, \quad |\uparrow\rangle = c_\uparrow^\dagger|0\rangle, \quad |\downarrow\rangle = c_\downarrow^\dagger|0\rangle, \quad |2\rangle = c_\uparrow^\dagger c_\downarrow^\dagger|0\rangle.$$

The sign convention of $|2\rangle$ determines the signs of the X-operators that involve it.
The operator is constructed as $X^{ab} = C_a\,(1-n_\uparrow)(1-n_\downarrow)\,C_b^\dagger$ with $C_0 = 1$, $C_\uparrow = c_\uparrow^\dagger$, $C_\downarrow = c_\downarrow^\dagger$, $C_2 = c_\uparrow^\dagger c_\downarrow^\dagger$, and normal ordered. For example $X^{0\uparrow} = c_\uparrow(1-n_\downarrow)$, $X^{\uparrow\uparrow} = n_\uparrow(1-n_\downarrow)$, $X^{\uparrow\downarrow} = c_\uparrow^\dagger c_\downarrow = S^+$, $X^{22} = n_\uparrow n_\downarrow$.
The X-operators satisfy $X^{ab}X^{cd} = \delta_{bc}X^{ad}$ and the completeness relation $\sum_a X^{aa} = 1$. X^{ab} is fermionic if the states a and b differ in the particle number by one.
An invalid state label gives the message `xoperator::state` and returns `$Failed`.
Abstract X-operators with the algebra built in are declared with `snegxoperators`. `xtofermions` and `fermionstox` convert between the abstract X-operators of the Hubbard atom and fermions, and `xtospin` converts the X-operators of singly occupied sites to spin operators.

Example.

```
snegfermionoperators[c];
xoperator[c[1], "0", "UP"]
```

xtofermions

`xtofermions[expr, X -> c]` replaces the X-operators `X[j..., a, b]` of the Hubbard atom in *expr* by their fermion representation `xoperator[c[j...], a, b]`. `X` must be declared with `snegxoperators[X, "HubbardAtom"]`, with the states `"0"`, `"UP"`, `"DO"`, `"2"`; otherwise the message `xtofermions::states` is issued and `$Failed` is returned. See `xoperator` for the state conventions.
The X-operators of the site `X[j...]` become functions of the fermion level `c[j...]`; the result is expanded and normal ordered. This allows exact diagonalization with the fermion bases `qszbasisvc`, ... and `makematricesbzvc`.
Within the full Fock space of the fermions, the abstract algebra of the X-operators is reproduced exactly: products, commutators, and the (anti)commutation with other fermions.
`fermionstox` is the inverse transformation.

Example.

```
snegfermionoperators[c];
snegxoperators[X, "HubbardAtom"];
xtofermions[X[1, "UP", "0"] + X[1, "2", "2"], X -> c]
```

xtospin

`xtospin[expr, X -> S]` replaces the X-operators of singly occupied sites, $X^{\sigma\sigma'}$ with $\sigma, \sigma' \in \{\uparrow, \downarrow\}$, by spin-1/2 operators `S`.
The replacement is

$$X^{\uparrow\uparrow} = \frac{1}{2} + S^z, \quad X^{\downarrow\downarrow} = \frac{1}{2} - S^z, \quad X^{\uparrow\downarrow} = S^+, \quad X^{\downarrow\uparrow} = S^-,$$

for each site `X[j..., a, b]` → `S[j..., t]`. It is valid in the subspace where the site is singly occupied ($X^{\uparrow\uparrow} + X^{\downarrow\downarrow} = 1$).
Terms that contain X-operators involving the states `"0"` or `"2"` are dropped, and the message `xtospin::out` lists them. If the expression is projected onto single occupancy (e.g. an effective Hamiltonian from `schriefferwolff` with `projector1`), no such terms occur.
`X` must be declared with `snegxoperators[X, "HubbardAtom"]` (message `xtofermions::states`), and `S` with `snegspinoperators` (message `xtospin::spinop`); otherwise `$Failed` is returned.
Use `fermionstox` first to convert fermion expressions to X-operators.

Example.

```
snegxoperators[X, "HubbardAtom"];
snegspinoperators[{S, 1/2}];
xtospin[X[1, "UP", "DO"] + X[1, "DO", "DO"], X -> S]
```

15.5 Sums, Lattices, and Momentum Space

blochhamiltonian

`blochhamiltonian[H, {ck[a], ck[b], ...}, k]` gives the Bloch Hamiltonian matrix $h(k)$ of a translation-invariant bilinear Hamiltonian $H = \sum_k \sum_{\alpha\beta} c^\dagger_{k\alpha} h_{\alpha\beta}(k) c_{k\beta}$ in momentum space.
H is a Hamiltonian written with momentum operators, typically the result of `fourier`, and k is a momentum index (`snegmomentumindexes`). The name of the summation index in H does not matter.
The orbitals are given as templates: `ck` or `ck[a, ...]`, where `a, ...` are orbital indexes. The momentum is inserted as the first index after `CR/AN` (option `"IndexPosition"`). For fermions, the spin index is appended as the last index and varies fastest: $\{\uparrow,\downarrow\}$ (`UP`, `DO`) for spin-1/2 operators, $S, S-1, \ldots, -S$ for operators of spin S (0 for spinless fermions). Boson orbitals have no spin index. The rows and columns of h are ordered accordingly, e.g. $(A\uparrow, A\downarrow, B\uparrow, B\downarrow)$.
The matrix elements are computed as $h_{\alpha\beta}(k) = \{c_{k\alpha}, [H, c^\dagger_{k\beta}]\}$ for fermions and $[b_{k\alpha}, [H, b^\dagger_{k\beta}]]$ for bosons, and simplified with `sumExpand`, `sumCollect`, and `sumSimplifyKD`. Constant terms of H do not contribute.
With `"Check" -> True` (default), the matrix elements between different momenta $k \neq p$ are computed; if they do not vanish, H is not translation invariant for the chosen unit cell (for example a staggered potential without a doubled cell), the message `blochhamiltonian::transl` is issued, and `$Failed` is returned. A non-Hermitian $h(k)$ gives the message `blochhamiltonian::herm`, and the matrix is returned.
Terms that do not conserve the particle number (pairing or linear terms) are not described by $h(k)$; they give the message `blochhamiltonian::anom` and `$Failed`. Use `bdghamiltonian` for the Bogoliubov-de Gennes form. Interaction terms leave operators in the matrix elements and give `blochhamiltonian::rest`.
The default convention is $c_{R\alpha} = N_s^{-1/2} \sum_k e^{ik\cdot R} c_{k\alpha}$, with the phase of the unit cell R for all orbitals in the cell (as in `fourier`). With `"OrbitalPositions" -> {tau_a, tau_b, ...}`, one position per orbital template, the matrix is returned in the convention $c_{R\alpha} = N_s^{-1/2} \sum_k e^{ik\cdot(R+\tau_\alpha)} c_{k\alpha}$, $h^{\mathrm{II}}_{\alpha\beta} = e^{-ik\cdot\tau_\alpha} h_{\alpha\beta} e^{ik\cdot\tau_\beta}$. Both matrices have the same eigenvalues; the second convention is needed for Berry phases and polarization.
The result contains Bloch phases `blochphase[k, X]`; `blochphaseexp` substitutes explicit components.
`"Check" True` whether to check translation invariance and Hermiticity
`"IndexPosition" 1` position of the momentum among the indexes after `CR/AN`
`"OrbitalPositions" None` positions τ_α of the orbitals in the unit cell

`blochhamiltonian::index` k is not a momentum index
`blochhamiltonian::orb` an orbital template is not a fermion or boson operator
`blochhamiltonian::anom` H contains terms that do not conserve the particle number
`blochhamiltonian::rest` a matrix element contains operators or unevaluated sums
`blochhamiltonian::transl` H is not translation invariant for the given orbitals
`blochhamiltonian::herm` $h(k)$ is not Hermitian (warning)

Example. Tight-binding chain, $\epsilon_k = -2t\cos k$ for both spin directions:

```
snegfermionoperators[c, ck];
snegrealconstants[t];
sneglatticeindexes[{R}, Ns];
snegmomentumindexes[{k}, Ns];
Hk = fourier[-t sum[hop[c[R], c[R + 1]], {R}], c -> ck, R -> k];
h = blochhamiltonian[Hk, {ck}, k]
```

blochphase

`blochphase[k, X]` represents the Bloch phase $e^{ik\cdot X}$ of the momentum k and the lattice vector X.
The first argument is a momentum, the second a lattice vector: a lattice index, a lattice vector declared with `sneglatticevectors`, a combination of them, or, in one dimension, an integer displacement. In output, `blochphase[k, X]` is shown as $e^{ik\cdot X}$.
Automatic simplifications:
`blochphase[0, X]` and `blochphase[k, 0]` are 1;
sums in the lattice argument are split, $e^{ik\cdot(R+a_1)} = e^{ik\cdot R}e^{ik\cdot a_1}$;
integer factors are moved to the momentum, `blochphase[k, 2 R]` is `blochphase[2 k, R]`, and integer displacements are written as `blochphase[n k, 1]`;
products with a common lattice argument are combined, $e^{ik\cdot X}e^{iq\cdot X} = e^{i(k+q)\cdot X}$, and integer powers give $e^{ink\cdot X}$;
`Conjugate[blochphase[k, X]]` is `blochphase[-k, X]`, and `conj` acts in the same way;
with explicit vectors of equal length, `blochphase[{k1, k2}, {x1, x2}]` is `Exp[I (k1 x1 + k2 x2)]`.
`blochphase` has the attribute `NumericFunction`, so Bloch phases are scalars for `nc` and `sum`.
`sumSimplifyKD` performs the lattice sums $\sum_R e^{iK\cdot R} = N_s\delta_{K,0}$ and $\sum_k e^{ik\cdot X} = N_s\delta_{X,0}$ when the summation index appears only in Bloch phases that multiply the summand.
`blochphaseexp` substitutes explicit components and converts the phases to exponentials. Symbolic momenta are abstract labels; the product $k\cdot X$ has no further

meaning until components are given.

Example.

```
snegmomentumindexes[{k, q}, Ns];
sneglatticeindexes[{R}, Ns];
blochphase[k, R]
```

blochphaseexp

`blochphaseexp[expr, rules]` applies the substitution *rules* (e.g. `{k -> {kx, ky}, a1 -> {1, 0}}`) to *expr* and converts the Bloch phases `blochphase[k, X]` to exponentials $e^{ik\cdot X}$.
`blochphaseexp[expr]` converts the Bloch phases without substitutions.
After the substitution, `blochphase[k, X]` becomes `Exp[I k . X]` if both arguments are lists (vectors of equal length are converted already by `blochphase`), and `Exp[I k X]` otherwise; the latter is appropriate in one dimension with integer displacements, `blochphase[k, 1]` → e^{ik}.
The components are given in any Cartesian or lattice basis, as long as momenta and lattice vectors are written in dual bases, so that $k \cdot X$ is the phase.
Use `ExpToTrig` and `Simplify` to obtain the dispersion in trigonometric form.

Example.

```
snegmomentumindexes[{k}, Ns];
ExpToTrig[blochphaseexp[blochphase[k, 1] + blochphase[-k, 1]]]
```

clusterhop

`clusterhop[T, c[a], d[b], delta, {L1, ...}]` gives the hopping $\sum_n c^\dagger_{na}\, T\, d_{n+\delta,b}$ + h.c. on a finite periodic cluster with $L_1 \times \cdots$ sites.
`clusterhop[T, c, d, delta, {L1, ...}]` is the same for operators without orbital indexes.
`clusterhop[T, c[a], d[b], delta, {L1, ...}, "Twist" -> theta]` uses twisted boundary conditions.
Sites are labeled as in `fourierrules`: integers $n = 0, \ldots, L-1$ in one dimension, integer vectors $n = (n_1, n_2, \ldots)$ in lattice coordinates otherwise. The displacement *delta* is an integer (1D) or an integer vector of the same dimension; otherwise `clusterhop::dims` is issued and `$Failed` returned. The target site $n + \delta$ is folded into the cluster modulo L_a.
The site label is inserted as the first index after `CR/AN`, before the orbital indexes a and b; for fermions, the spin index is appended last: `c[a]` gives `c[CR, n, a, σ]`.
T is a number or a symbol (times the identity in spin space), or a matrix in the spin indexes, e.g. a 2×2 matrix for spin-1/2 fermions with rows and columns ordered

(↑, ↓), as in `matrixhop`. `c` and `d` can be fermions (with spin), spinless fermions, or bosons.
With `"Twist" -> theta`, the bonds that cross the boundary in direction a are multiplied by $e^{i\theta_a w_a}$, where w_a is the number of windings, corresponding to $c_{n+L_a e_a} = e^{i\theta_a} c_n$. A scalar twist applies to all directions. A twist describes a magnetic flux θ through a ring and is used for twist-averaged boundary conditions. The default `Automatic` means periodic boundary conditions.
Every site contributes one bond, so on a ring with $L = 2$ the bonds to the left and to the right neighbor coincide and the hopping is doubled; this is consistent with the band energies $-2t \cos k$ at $k = 0, \pi$. Likewise, for $L_a = 1$ the hopping in direction a becomes an on-site term.
`"Twist" Automatic` twist angles θ_a of the boundary conditions

Example. Hopping on a three-site ring of spinless fermions:

```
snegspinlessfermionoperators[f];
snegrealconstants[t];
clusterhop[-t, f, f, 1, {3}]
```

FactorSeparableSumCustom

`FactorSeparableSumCustom[sum[expr, {i, j, ...}]]` factorizes a symbolic sum into a product of independent sums, $\sum_{ijk} f_i g_j h_{ik} = (\sum_{ik} f_i h_{ik})(\sum_j g_j)$.
The factors of the summand (the arguments of `Times`) and the summation indexes form a bipartite graph, in which a factor is connected with the indexes on which it depends. Each connected component gives one factor of the result: factors without indexes are moved out of the sum, indexes that do not appear in the summand give `sum[1, {i}]`, and all other components give a sum of the product of their factors over their indexes.
The summand is not expanded; a summand that is a sum of terms is a single factor.
`rulesumFactor` applies `FactorSeparableSumCustom` to all sums in an expression.

Example.

```
snegfreeindexes[i, j, k];
snegrealfunctions[f, g, h];
FactorSeparableSumCustom[sum[f[i] g[j] h[i, k], {i, j, k}]]
```

fourier

`fourier[expr, c -> ck, R -> k]` transforms the site operators `c` in *expr* to the momentum operators `ck`, $c_R = N_s^{-1/2} \sum_k e^{ik\cdot R} c_k$.
`fourier[expr, c -> ck, R -> k, "Inverse" -> True]` transforms the momentum operators `ck` back to the site operators `c`, $c_k = N_s^{-1/2} \sum_R e^{-ik\cdot R} c_R$.

R must be a lattice index (`sneglatticeindexes`) and k a momentum index (`snegmomentumindexes`); N_s is `latticesize[k]` (`latticesize[R]` for the inverse transformation). `c` and `ck` are fermion or boson operators.
The convention is

$$c_{R\alpha\sigma} = \frac{1}{\sqrt{N_s}} \sum_k e^{ik\cdot R} c_{k\alpha\sigma}, \qquad c^\dagger_{R\alpha\sigma} = \frac{1}{\sqrt{N_s}} \sum_k e^{-ik\cdot R} c^\dagger_{k\alpha\sigma},$$

so that $\sum_R e^{iK\cdot R} = N_s \delta_{K,0}$ and $\sum_k e^{ik\cdot X} = N_s \delta_{X,0}$. The transformation is unitary: canonical (anti)commutation relations and $\sum_R n_R = \sum_k n_k$ are preserved.
The site or momentum label is the first index after `CR/AN` (option `"IndexPosition"`). Orbital indexes and the spin index are kept: `c[CR, R + a1, "A", UP]` becomes a sum over `ck[CR, k, "A", UP]` with the Bloch phase `blochphase[-k, R + a1]`.
Every operator `c[CR | AN, ...]` is transformed, whatever its site label: `c[0]`, `c[R + 1]`, `c[R + a1]`, or a free label.
Bloch phases are represented by `blochphase[k, X]`, $e^{ik\cdot X}$. Phases with a common lattice argument are combined and sums in the lattice argument are split, so $e^{-ik\cdot R} e^{ik'\cdot(R+1)} = e^{i(k'-k)\cdot R} e^{ik'}$.
With `"Simplify" -> True` (default), the result is simplified by repeated `sumExpand`, `Expand`, `sumSimplifyKD`, and `sumCollect`: the lattice sums over R are performed, which gives momentum conservation, and the Kronecker deltas are eliminated. Automatically renamed dummy indexes with the root k (`k1`, ...) are renamed back to k where possible.
To transform several operator families, call `fourier` repeatedly with different momentum indexes, e.g. `k` for electrons and `q` for phonons.
The result is a sum over momenta; `blochhamiltonian` extracts the Bloch matrix $h(k)$ of a bilinear Hamiltonian, and `sumCluster` specializes the result to a finite periodic cluster.

`"IndexPosition" 1` position of the site or momentum label among the indexes after `CR/AN`

`"Inverse" False` whether to transform the momentum operators `ck` to the site operators `c`

`"Simplify" True` whether to perform the lattice sums and simplify the result

`fourier::index` R is not a lattice index or k is not a momentum index; returns `$Failed`

`fourier::op` `c` (`ck` for the inverse) is not a fermion or boson operator; returns `$Failed`

`fourier::capture` *expr* already contains k (R for the inverse), which would be captured by the new sum; returns `$Failed`

Example. Hubbard chain, $H = -t \sum_{R\sigma} (c^\dagger_{R\sigma} c_{R+1,\sigma} + \text{h.c.}) + U \sum_R n_{R\uparrow} n_{R\downarrow}$:

```
snegfermionoperators[c, ck];
```

```
snegrealconstants[t, U];
sneglatticeindexes[{R}, Ns];
snegmomentumindexes[{k}, Ns];
H = -t sum[hop[c[R], c[R + 1]], {R}] + U sum[hubbard[c[R]], {R}];
fourier[H, c -> ck, R -> k]
```

fourierrules

`fourierrules[c -> ck, {L1, L2, ...}]` gives replacement rules that express the site operators `c` of a finite periodic $L_1 \times L_2 \times \cdots$ cluster by the momentum operators `ck`.

`fourierrules[c -> ck, {L1, ...}, "Twist" -> theta]` uses twisted boundary conditions.

Sites are integer vectors in lattice coordinates, $n = (n_1, n_2, \ldots)$ with $n_a = 0, \ldots, L_a - 1$; in one dimension they are plain integers $n = 0, \ldots, L - 1$. Momentum labels are integer vectors m of the same form, with the crystal momenta $k_a = (2\pi m_a + \theta_a)/L_a$. The rules implement the unitary transformation

$$c_n = \frac{1}{\sqrt{N}} \sum_m e^{ik_m \cdot n} c_m, \qquad c_n^\dagger = \frac{1}{\sqrt{N}} \sum_m e^{-ik_m \cdot n} c_m^\dagger, \qquad N = L_1 L_2 \cdots,$$

the finite-cluster version of the convention of `fourier`, with $k_m \cdot n = \sum_a (2\pi m_a + \theta_a) n_a / L_a$.

The site label is the first index after `CR/AN` (option `"IndexPosition"`); orbital and spin indexes are kept. Only operators whose site label is a site of the cluster are replaced; other labels, such as $n = L$, are left unchanged.

Twisted boundary conditions $c_{n+L_a e_a} = e^{i\theta_a} c_n$ shift the allowed momenta by θ_a / L_a. `"Twist" -> theta` applies the same twist in all directions, `"Twist" -> {theta1, theta2, ...}` one twist per direction. The default `Automatic` means periodic boundary conditions, $\theta = 0$. Use `clusterhop` with the same twist for the real-space Hamiltonian.

For numerical twists, the unitarity of the transformation matrix is checked; a failure gives the message `fourierrules::unitary`.

Apply the rules with `/.` and `Expand` the result.

`"IndexPosition" 1` position of the site label among the indexes after `CR/AN`

`"Twist" Automatic` twist angles θ_a of the boundary conditions

Example. A site operator of a three-site ring:

```
snegspinlessfermionoperators[f, fk];
f[AN, 1, 0] /. fourierrules[f -> fk, {3}]
```

FullSimplifyKD

`FullSimplifyKD[expr]` simplifies expressions with Kronecker deltas and step functions by `FullSimplify` with the identities used by `SimplifyKD`.
`FullSimplifyKD` applies the same identities as `SimplifyKD` ($\delta^n = \delta$, $\theta(k) + \theta(-k) = 1$, $\theta(k)\theta(-k) = 0$, $\delta_{kl}\theta(k)\theta(-l) = 0$, ...), but within `FullSimplify`, which tries more transformations and is slower.
As for `SimplifyKD`, the arguments of the step functions are assumed to be nonzero.

Example.

```
FullSimplifyKD[KroneckerDelta[i, j]^4 + UnitStep[k] + UnitStep[-k]]
```

latticeindexQ

`latticeindexQ[R]` gives `True` if R is a symbol declared as a lattice index with `sneglatticeindexes`, and `False` otherwise.
`latticeindexQ[R]` is `latticeindextype[R] === "Lattice"`. Lattice vectors (`sneglatticevectors`), momentum indexes, and expressions such as `R + 1` give `False`.

Example.

```
sneglatticeindexes[{R}, Ns];
snegmomentumindexes[{k}, Ns];
sneglatticevectors[a1];
{latticeindexQ[R], latticeindexQ[k], latticeindexQ[a1], latticeindexQ[R
    + 1]}
```

latticeindextype

`latticeindextype[x]` gives `"Lattice"` for a lattice index, `"Momentum"` for a momentum index, and `None` for all other expressions.
The type is set by `sneglatticeindexes` and `snegmomentumindexes` as an upvalue of the index symbol, and copied to indexes created by the automatic renaming of dummy indexes.
Lattice vectors (`sneglatticevectors`) have the type `None`, although they count as lattice quantities in Kronecker deltas and shifts.

Example.

```
sneglatticeindexes[{R}, Ns];
snegmomentumindexes[{k}, Ns];
sneglatticevectors[a1];
snegfreeindexes[i];
latticeindextype /@ {R, k, a1, i}
```

latticesize

`latticesize[k]` gives the number of unit cells N_s of the lattice of a declared lattice or momentum index k.
The value is the second argument of `sneglatticeindexes` or `snegmomentumindexes`.
For other symbols, `latticesize[k]` remains unevaluated.
N_s is the number of values of the index: `sum[1, {k}]` is `latticesize[k]`.

Example.

```
sneglatticeindexes[{R}, Ns];
snegmomentumindexes[{k}, 8];
{latticesize[R], latticesize[k], latticesize[x]}
```

listlatticeindexes

`listlatticeindexes` is the list of the symbols declared as lattice indexes.
Symbols are added by `sneglatticeindexes` and by the automatic renaming of dummy lattice indexes in products of sums. The list is sorted.
The list is for inspection; the index properties are stored with the symbols, so removing a symbol from the list does not undo its declaration.

Example.

```
sneglatticeindexes[{R, Rp}, Ns];
listlatticeindexes
```

listlatticevectors

`listlatticevectors` is the list of the symbols declared as lattice vectors with `sneglatticevectors`.
The list is sorted and is for inspection; the properties are stored with the symbols.

Example.

```
sneglatticevectors[a1, a2];
listlatticevectors
```

listmomentumindexes

`listmomentumindexes` is the list of the symbols declared as momentum indexes.
Symbols are added by `snegmomentumindexes` and by the automatic renaming of dummy momentum indexes in products of sums. The list is sorted.
The list is for inspection; the index properties are stored with the symbols.

Example.

```
snegfermionoperators[ck];
snegmomentumindexes[{k, q}, Ns];
listmomentumindexes
```

momentumbasis

`momentumbasis[bz, dims]` splits each sector of a momentum-orbital basis by its total crystal momentum and appends the momentum label to the quantum numbers. For creation operators with momentum labels m_j the total label is $\sum_j m_j$ modulo the side length along each direction; the empty state has zero total momentum. These are grid labels, not angles.
Input states must already have definite momentum. This function groups them; it does not Fourier transform real-space states. A linear combination of states with different momenta gives a message and `$Failed`.
For occupation vectors, retain the single-particle `BASIS` belonging to the input. `"IndexPosition" 1` position of the momentum label after the operator type

Example. On a two-site ring with one electron of each spin the pairs with both electrons at the same momentum have $K = 0$. A pair with one electron at each momentum has $K = 1$:

```
snegfermionoperators[f];
b = momentumbasis[qszbasis[{f[0],f[1]}],{2}];
TableForm[{Last[First[#]],Length[Last[#]]} & /@ Select[b,Take[First
    [#],2] == {0,0} &],
 TableHeadings -> {None,{"K","States"}}]
```

momentumindexQ

`momentumindexQ[k]` gives `True` if k is a symbol declared as a momentum index with `snegmomentumindexes`, and `False` otherwise.
`momentumindexQ[k]` is `latticeindextype[k] === "Momentum"`. Expressions such as `k + q` give `False`.

Example.

```
sneglatticeindexes[{R}, Ns];
snegmomentumindexes[{k, q}, Ns];
{momentumindexQ[k], momentumindexQ[R], momentumindexQ[k + q]}
```

ruleExpandUnderSum

`expr /. ruleExpandUnderSum` expands the summands of the symbolic sums in *expr* with `Expand`.

`ruleExpandUnderSum` is `{sum[a_, it_List] :> sum[Expand[a], it]}`. The sums are not distributed over the resulting terms; combine with `rulesumDistribute` for that.
`Expand` does not expand denominators, in contrast to `ExpandAll`, which is used by `sumExpand`.

Example.

```
snegfreeindexes[k];
snegrealfunctions[f, g, h];
s = sum[(f[k] + g[k])/(1 + h[k])^2, {k}];
s /. ruleExpandUnderSum
```

ruleFullSimplifyUnderSum

`expr /. ruleFullSimplifyUnderSum` applies `FullSimplify` to the summands of the symbolic sums in *expr*.
`ruleFullSimplifyUnderSum` is `{sum[a_, {q__}] :> sum[FullSimplify[a], {q}]}`. Only the summands are simplified; the sums themselves are not expanded, collected, or combined, in contrast to `sumFullSimplify`.
With `/.`, only the outermost sums are matched; use `//.` for nested sums.

Example.

```
snegfermionoperators[c];
snegfreeindexes[k];
snegrealfunctions[ε];
s = sum[Sin[ε[k]]^2 c[CR, k, UP] + Cos[ε[k]]^2 c[CR, k, UP], {k}]
```

rulestripscalar

`expr /. rulestripscalar` removes the scalar markers `scalar[x]` from *expr*, replacing them by *x*.
`scalar[x]` is a marker that declares *x* as a scalar (non-operator) quantity. It is moved in front of `nc` products, absorbs other scalar factors (`a scalar[x]` becomes `scalar[a x]`), and adds up, `scalar[x] + scalar[y]` = `scalar[x + y]`. It is used for speed in expressions with symbolic sums; `sumSimplifyKD` eliminates Kronecker deltas also inside `scalar[...]`.
`rulestripscalar` is `{scalar[x_] :> x}`.

Example.

```
snegfermionoperators[c];
snegfreeindexes[k, p];
snegrealfunctions[ε];
```

```
e = sumSimplifyKD[sum[scalar[ε[k] KroneckerDelta[k, p]] c[CR, k, UP], {k
    }]]
```

rulesumDistribute

`expr /. rulesumDistribute` distributes the symbolic sums in *expr* over the terms of their summands, $\sum_k (a_k + b_k) \to \sum_k a_k + \sum_k b_k$, without expanding the summands.
`rulesumDistribute` is `{sum[a_ + b_, it_List] :> sum[a, it] + sum[b, it]}`.
Use `//.` to distribute completely; `/.` splits off one term at each level.
`sumExpand` also expands the summands with `ExpandAll`; `rulesumDistribute` acts only on summands that are already sums.

Example.

```
snegfreeindexes[k];
snegrealfunctions[f, g, h];
s = sum[f[k] h[k] + g[k] (1 + h[k]), {k}];
s //. rulesumDistribute
```

rulesumFactor

`expr /. rulesumFactor` factorizes the sums in *expr* into products of independent sums, $\sum_{ij} f_i g_j = \sum_i f_i \sum_j g_j$.
`rulesumFactor` is `{s:sum[_, _List] :> FactorSeparableSumCustom[s]}`; see `FactorSeparableSumCustom` for the factorization.
The summand must be a product of factors that depend on disjoint sets of indexes. A summand that is a sum of terms is not factorized; apply `sumExpand` first.

Example.

```
snegfreeindexes[i, j];
snegrealfunctions[f, g];
sum[3 f[i] g[j], {i, j}] /. rulesumFactor
```

SimplifyKD

`SimplifyKD[expr]` simplifies expressions with Kronecker deltas `KroneckerDelta` and step functions `UnitStep` in *expr*.
`SimplifyKD` calls `Simplify` with an additional transformation function that applies the identities
$\delta^n = \delta$ and $\theta^n = \theta$ for positive integer powers n;
$\theta(k) + \theta(-k) = 1, 1 - \theta(-k) = \theta(k), \theta(-k) - 1 = -\theta(k), \theta(k)\theta(-k) = 0$;
$\delta_{kl}\,\theta(k)\theta(-l) = \delta_{kl}\,\theta(-k)\theta(l) = 0$.

Here $\theta(x)$ is `UnitStep[x]`. With `SEA` ordering (see `ordering`), momenta are measured from the Fermi level, and $\theta(-k)$ and $\theta(k)$ are the occupation of the state k in the Fermi sea and its complement.
Since `UnitStep[0]` is 1, the identities with $\theta(k)$ and $\theta(-k)$ hold only for $k \neq 0$; `SimplifyKD` assumes that the arguments of the step functions are nonzero.
`SimplifyKD` does not perform sums; use `sumSimplifyKD`. `FullSimplifyKD` uses `FullSimplify` instead of `Simplify`.

Example.

```
{SimplifyKD[UnitStep[k] + UnitStep[-k]], SimplifyKD[1 - UnitStep[-k]],
    SimplifyKD[KroneckerDelta[a, b]^2]}
```

sneglatticeindexes

`sneglatticeindexes[{R1, R2, ...}, Ns]` declares *R1*, *R2*, ... as indexes that label the unit cells of a periodic lattice with *Ns* unit cells.
A lattice index R labels the unit cells (Bravais lattice vectors) of a lattice with periodic boundary conditions. Site labels are defined modulo the lattice period, so the shifts $R \to \pm R + R_0$ are bijections of the index set. SNEG uses this to solve Kronecker deltas such as $\sum_R \delta_{R',R+1} f(R) = f(R'-1)$, to shift summation variables (`sumShift`), and to perform lattice sums $\sum_R e^{iK\cdot R} = N_s \delta_{K,0}$ (`sumSimplifyKD`).
In one dimension, displacements can be written with integers, `c[R + 1]`. In higher dimensions, declare the primitive vectors with `sneglatticevectors` and write `c[R + a1]`. The site index is the first index after `CR/AN` of an operator; orbital indexes follow, and the spin is last: `c[CR, R, "A", UP]`.
The declaration makes each index a free index (`snegfreeindexes`) and a real quantity (`Conjugate[R]` is `R`), sets `latticeindextype[R]` to `"Lattice"` and `latticesize[R]` to *Ns*, and appends the index to `listlatticeindexes`.
Ns is a positive integer or an expression built from symbols, such as `Ns` or `L1 L2`; the symbols are declared real constants. Other values give the message `sneglatticeindexes::size`, and nothing is declared.
A sum over a lattice index that does not appear in the summand evaluates to *Ns*: `sum[1, {R}]` is *Ns*.
Indexes created by the automatic renaming of dummy indexes (`R1`, `R2`, ...) inherit the declaration.
Generic indexes (`snegfreeindexes`) are not assumed to be periodic: for them, deltas of linear combinations are not solved, because on an open chain $\sum_i \delta_{j,i+1} f(i) = f(j-1)$ fails at the boundary.
Lattice and momentum indexes with the same *Ns* describe the same lattice.
Momentum indexes are declared with `snegmomentumindexes`.
A repeated declaration overwrites the size.

Example.

```
sneglatticeindexes[{R}, Ns];
{latticeindexQ[R], latticesize[R], Conjugate[R], sum[1, {R}]}
```

sneglatticevectors

`sneglatticevectors[a1, a2, ...]` declares *a1*, *a2*, ... as lattice vectors that can be used as displacements of lattice indexes, as in `c[R + a1]`.
Lattice vectors are symbolic displacements such as the primitive vectors $\mathbf{a}_1, \mathbf{a}_2$ of a two-dimensional lattice. They are declared real constants and appended to `listlatticevectors`. They are not summation indexes: `latticeindexQ[a1]` is `False`.
In combinations with lattice indexes, lattice vectors count as lattice quantities: `sumSimplifyKD` solves $\sum_R \delta_{R',R+a_1} f(R) = f(R' - a_1)$, and `sumShift` accepts shifts $R \to R + a_1$.
Bloch phases $e^{ik\cdot a_1}$ appear as `blochphase[k, a1]`. Explicit components are substituted with `blochphaseexp`, for example `blochphaseexp[h, {k -> {kx, ky}, a1 -> {1, 0}, a2 -> {0, 1}}]`. On finite clusters, `sumCluster` takes the components in lattice coordinates.
In one dimension, integer displacements `c[R + 1]` can be used without declaring lattice vectors.
A lattice vector is a symbol that may be zero for SNEG: $\delta_{R+a_1,R}$ is reduced to $\delta_{a_1,0}$, not to 0.

Example.

```
sneglatticevectors[a1, a2];
{Conjugate[a1], latticeindexQ[a1], listlatticevectors}
```

snegmomentumindexes

`snegmomentumindexes[{k1, k2, ...}, Ns]` declares *k1*, *k2*, ... as indexes that label the crystal momenta of a periodic lattice with *Ns* unit cells.
A momentum index k labels the N_s crystal momenta in the Brillouin zone. Momenta are defined modulo reciprocal-lattice vectors, so Umklapp processes are included implicitly and the shifts $k \to \pm k + q$ are bijections of the index set. SNEG uses this to solve Kronecker deltas such as $\sum_k \delta_{p,k+q} f(k) = f(p - q)$ (momentum conservation), to shift summation variables (`sumShift`, `sumMomentumCanonical`), and to perform the sums $\sum_k e^{ik\cdot X} = N_s \delta_{X,0}$ (`sumSimplifyKD`).
Symbolic momenta are abstract labels; in more than one dimension they stand for vectors. Explicit components are substituted only when Bloch phases are evaluated with `blochphaseexp`, and integer labels on finite clusters are used by `sumCluster`.

The declaration makes each index a free index (`snegfreeindexes`) and a real quantity, sets `latticeindextype[k]` to `"Momentum"` and `latticesize[k]` to *Ns*, and appends the index to `listmomentumindexes`. *Ns* is a positive integer or an expression built from symbols, as for `sneglatticeindexes`; invalid sizes give the message `sneglatticeindexes::size`.
`sum[1, {k}]` is *Ns*. Indexes created by the automatic renaming of dummy indexes (`k1`, `k2`, ...) inherit the declaration.
The momentum label is the first index after `CR`/`AN` of an operator: `ck[CR, k, UP]`, `fk[AN, k, "A", 0]`.
Deltas whose arguments mix momenta and lattice quantities are not solved, and coefficients other than ± 1 are not solved ($\delta_{2k,q}$ has several solutions).

Example.

```
snegmomentumindexes[{k, p, q}, Ns];
{momentumindexQ[k], latticesize[k], sum[1, {k}]}
```

sum

`sum[expr, {i, j, ...}]` represents the symbolic sum $\sum_{i,j,\ldots}$ expr over the dummy indexes *i*, *j*, ...
`sum[expr, {}]` is *expr*.
`sum` is a formal sum: the range of the indexes is not specified and the sum is never evaluated term by term. Typical uses are band Hamiltonians $H = \sum_{k\sigma} \epsilon_k c^\dagger_{k\sigma} c_{k\sigma}$, lattice Hamiltonians $\sum_R (\ldots)$, and operators such as the density $\rho_q = \sum_k c^\dagger_{k+q} c_k$.
The indexes should be declared, so that SNEG treats them as scalars and not as unknown operators. The declaration determines the index type and the value of `sum[1, {i}]`, the number of values of the index:
`snegfreeindexes[i]` generic index; `sum[1, {i}]` stays unevaluated
`sneglatticeindexes[{R}, Ns]` unit cells of a periodic lattice; `sum[1, {R}]` is N_s
`snegmomentumindexes[{k}, Ns]` crystal momenta; `sum[1, {k}]` is N_s
`snegspincomponentindexes[a]` Cartesian spin components; `sum[1, {a}]` is 3
Kronecker deltas of linear combinations such as $\delta_{p,k+q}$ are solved, and lattice sums of Bloch phases are performed, only for lattice and momentum indexes, see `sumSimplifyKD` and `fourier`.
The index list is sorted, so `sum[a, {k, i}]` is `sum[a, {i, k}]`.
Factors that do not depend on the summation indexes are moved in front of the sum; a summand *z* free of all indexes gives `z sum[1, {...}]`.
Nested sums are merged, `sum[sum[a, {i}], {j}]` is `sum[a, {i, j}]`.
In `nc`, a sum is moved to the front when the other factors do not contain its indexes: $c_1^\dagger \sum_k c_k = \sum_k c_1^\dagger c_k$.

A product of sums in `nc` is combined into a single sum. Dummy indexes that occur in both factors are renamed: the second `k` becomes `k1`, `k2`, ... (the first name that is not among the indexes of the two sums and does not occur in the expression). The new index is declared with `snegfreeindexes` and inherits the properties of the original: realness, lattice or momentum type and size, spin-component type.
`komutator[sum[a, ...], sum[b, ...]]` renames the indexes of the second argument in both terms of $ab - ba$.
`conj[sum[a, {...}]]` is `sum[conj[a], {...}]`.
`sum` does not distribute over `Plus` automatically: `sum[a + b, {k}]` and `sum[a, {k}] + sum[b, {k}]` are different expressions. Use `sumExpand` to distribute and expand the summand, `sumCollect` to combine sums, `sumSimplifyKD` to perform sums over Kronecker deltas, and `sumFullSimplify` for a heuristic combination of these steps.
`sumThread` threads a sum over a list.
`sum[expr, {}]` is *expr*; a sum without indexes is the summand.
Sums that contain only scalars are scalars (`isnumericQ` gives `True`), so they can be moved out of `nc` products.
The automatic renaming is controlled by the global variable `sumAutoRename` (default `True`). With `sumAutoRename = False`, colliding indexes are not renamed and the message `snegsumJoin::overlap` is issued.

Example. Kinetic energy of a band of electrons, $H = \sum_{k\sigma} \epsilon_k c^\dagger_{k\sigma} c_{k\sigma}$:

```
snegfermionoperators[c];
snegfreeindexes[k, sigma];
snegrealfunctions[ε];
H = sum[ε[k] number[c[k], sigma], {k, sigma}]
```

sum2list

`sum2list[a + b + ...]` gives the list of terms `{a, b, ...}`.
`sum2list[x]` gives `{x}` for an expression x that is not a sum.
`sum2list` splits an expression with head `Plus` into its terms. It does not act on symbolic sums `sum[...]`: a symbolic sum is a single term.
Apply `Expand` first to obtain the individual operator monomials of an expression.

Example.

```
sum2list[a + b + c]
```

sumAbstractIndex

`sumAbstractIndex[expr, idx]` replaces the summation indexes of every sum in *expr* by the abstract indexes `idx[1]`, `idx[2]`, ..., numbered in the order of the index list of each sum.

Dummy indexes can be renamed freely. After `sumAbstractIndex`, sums that differ only in the names of their dummy indexes, such as $\sum_k f_k$ and $\sum_p f_p$, have identical forms and are combined by `Plus` automatically or by `sumCollect`.
Each sum is renamed separately: the indexes of a sum over $\{i, j\}$ become `idx[1]` and `idx[2]` in the order of the sorted index list.
`idx` is a symbol; it is given the attribute `NumericFunction`, so that `idx[n]` is treated as a scalar.
`sumNameIndex[expr, idx, {i, j, ...}]` replaces `idx[1]`, `idx[2]`, ... by ordinary index names.
The renaming does not change the summand otherwise. Sums whose summands differ by a permutation of the dummy indexes, such as $\sum_{ij} f_i g_j$ and $\sum_{ij} f_j g_i$, are not identified. Use this function for specific simplifications; the order of the indexes in each sum determines the result.

Example.

```
snegfreeindexes[i, j];
snegrealfunctions[f];
sumAbstractIndex[sum[f[i, j], {i, j}], idx]
```

sumAutoRename

`sumAutoRename` is a global variable that controls the automatic renaming of dummy indexes in products of symbolic sums. The default is `True`.
With `sumAutoRename = True`, a product `nc[sum[a, {k}], sum[b, {k}]]` is combined into one sum in which the second dummy index is renamed to `k1` (or `k2`, ...), the first name that is not among the indexes of the two sums and does not occur in the expression. The new index is declared with `snegfreeindexes` and inherits the properties of `k` (realness, lattice or momentum type, spin-component type).
`komutator` of two sums renames the indexes of the second sum in both products.
With `sumAutoRename = True`, nested sums inside a sum of terms, `sum[sum[a, {i}] + c, {j}]`, are kept nested so that the renaming rules see the full expression; use `sumExpand` to distribute.
With `sumAutoRename = False`, the index lists are joined without renaming. If they overlap, the message `snegsumJoin::overlap` is issued and the result contains the same index twice, which is wrong; nested sums inside a sum of terms are distributed automatically.
Use `Block[{sumAutoRename = False}, ...]` to change the setting temporarily.

Example.

```
sumAutoRename
```

sumCluster

`sumCluster[expr, dims]` evaluates typed lattice and momentum sums on a finite periodic cluster with side lengths *dims*.
`sumCluster[expr, dims, rules]` substitutes the lattice vectors specified in *rules*.
Sites and momenta have integer labels starting at zero. Indices are folded modulo the cluster dimensions before evaluating operator products and momentum deltas.
In dimension greater than one supply vectors in lattice coordinates.
Symbolic lattice sizes become the product of the side lengths. Do not use a conflicting declared numeric size. Untyped dummy indices remain symbolic.
Only periodic boundary conditions are supported. For a twist, build the bonds with `clusterhop` and transform with `fourierrules`.
`"IndexPosition" 1` position of the lattice label after the operator type
`"Operators" Automatic` operator heads whose labels are folded, inferred by default

Example. The term crossing the boundary of a three-site ring is included once:

```
snegspinlessfermionoperators[f]; sneglatticeindexes[{R}, Ns];
ring = sumCluster[sum[hop[f[R], f[R+1]], {R}], {3}];
Expand[ring - clusterhop[1, f, f, 1, {3}]]
```

sumCollect

`sumCollect[expr]` combines symbolic sums over common indexes into single sums, $\sum_k a_k + \sum_k b_k \to \sum_k (a_k + b_k)$.
`sumCollect` applies two rules repeatedly to pairs of terms $z_1 \sum a_1 + z_2 \sum a_2$ whose prefactors $z_{1,2}$ do not depend on the summation indexes:
If both sums run over the same indexes, they are combined into $\sum (z_1 a_1 + z_2 a_2)$.
If the index lists overlap partially, the common indexes form the outer sum: $\sum_{ij} a_{ij} + \sum_i b_i \to \sum_i (\sum_j a_{ij} + b_i)$.
Sums with different names of the dummy indexes are not combined, because `sumCollect` does not rename indexes. Use `sumMomentumCanonical` (lattice and momentum indexes) or `sumAbstractIndex` and `sumNameIndex` to bring the dummy indexes to common names first.
Terms without sums are left outside the sums.
`sumExpand` is the inverse operation. `sumFullSimplify` combines `sumCollect` with other simplification steps.

Example.

```
snegfreeindexes[k];
snegrealfunctions[f, g];
snegrealconstants[x];
sumCollect[2 sum[f[k], {k}] - x sum[g[k], {k}]]
```

sumCollectSimplifyKD

`sumCollectSimplifyKD[expr]` collects the summand of every sum in *expr* with respect to Kronecker deltas and then applies `sumSimplifyKD`.

The summands are rewritten with `Collect[a, KroneckerDelta[__]]`, so that all terms with the same delta are combined into a single coefficient, $\delta_{ij}a_i + \delta_{ij}b_i \to \delta_{ij}(a_i + b_i)$, and products such as $\delta_{ij}(a_i + b_i)c_i$ are written as a delta times a coefficient. The deltas are then eliminated by `sumSimplifyKD`.

Compared with `sumSimplifyKD[sumExpand[expr]]`, the coefficients of the deltas are not expanded.

Example.

```
snegfreeindexes[i, j];
snegrealfunctions[f, g, h];
sumCollectSimplifyKD[sum[(f[i] + g[i]) h[i] KroneckerDelta[i, j], {i}]]
```

sumExpand

`sumExpand[expr]` distributes the symbolic sums in *expr* over the terms of their summands, $\sum_k (a_k + b_k) \to \sum_k a_k + \sum_k b_k$, and expands the summands with `ExpandAll`.

`sumExpand` applies the rules `sum[a + b, it] -> sum[a, it] + sum[b, it]` and `sum[a, it] -> sum[ExpandAll[a], it]` repeatedly, so nested sums are expanded as well. Factors free of the summation indexes are then moved out of the sums automatically.

Expressions outside sums are not expanded; combine with `Expand` where needed.

Most results of `komutator`, `nc`, and `vev` with sums must be expanded with `sumExpand` before `sumSimplifyKD` can eliminate Kronecker deltas term by term.

`sumCollect` is the inverse operation.

`ExpandAll` also expands denominators. To expand only the numerators, use `expr /. ruleExpandUnderSum`; to distribute without expanding, use `expr /. rulesumDistribute`.

Example.

```
snegfermionoperators[c];
snegfreeindexes[k];
snegrealfunctions[f];
s = sum[f[k] (number[c[k]] - 1), {k}]
```

sumFullSimplify

sumFullSimplify[expr] simplifies an expression with symbolic sums by FullSimplify with additional transformation functions for sums.
The transformation functions passed to FullSimplify are:
a fixed sequence of Expand, sumCollect, Simplify, sumSimplifyKD, SimplifyKD, sumExpand, and sumSimplify steps;
combinations of expanding, collecting, and simplifying used by sumSimplify, SimplifyKD, and the individual rules of sumExpand, sumCollect, and sumSimplifyKD;
sumCollect[sumMomentumCanonical[sumExpand[#]]] &, which shifts lattice and momentum summation indexes to a canonical form and renames automatically renamed dummy indexes (k1, ...) back to their root names, so that equivalent sums cancel.
FullSimplify keeps the form with the smallest complexity (by default, essentially the leaf count). The result can therefore be collected or expanded, and is not a canonical form. Two expressions are equal if sumFullSimplify of their difference gives 0; a nonzero result does not prove that they differ.
The momentum shifts are exact only for declared lattice and momentum indexes, see sumMomentumCanonical. Sums over generic indexes with different dummy names are not identified.
sumFullSimplify can be slow for large expressions. sumSimplify uses Simplify with a subset of the transformations; for predictable results, apply sumExpand, sumSimplifyKD, and sumCollect explicitly.

Example. Commutator of the particle number with the band Hamiltonian, $[N, H] = 0$:

```
snegfermionoperators[c];
snegfreeindexes[k];
snegrealfunctions[ε];
H = sum[ε[k] number[c[k]], {k}];
c1 = komutator[sum[number[c[k]], {k}], H];
sumFullSimplify[c1]
```

sumMomentumCanonical

sumMomentumCanonical[expr] shifts the summed lattice and momentum indexes in every sum of *expr* so that the first operator argument that depends on each index is the bare index, and renames automatically renamed dummy indexes back to their root names.
For each summed lattice or momentum index *m*, the operator arguments in the summand (all indexes after the first, CR/AN, index) are searched in order; the first

argument $e = c\,m + r$ that depends on m, with $c = \pm 1$, is made the bare index by the shift $m \to c\,(m - r)$, applied to the whole summand. Example: $\sum_k c^\dagger_{k+q} c_k \to \sum_k c^\dagger_k c_{k-q}$.

Dummy indexes created by the automatic renaming of `sum` (`k1`, `k2`, ... for the root `k`) are renamed to the root name if the root is a declared index of the same type that does not occur in the sum.

After this step, equivalent sums often have identical summands, so that they cancel or can be combined by `sumCollect`. `sumFullSimplify` uses `sumCollect[ sumMomentumCanonical[sumExpand[expr]]]` as one of its transformations.

The shifts are exact for lattice and momentum indexes, see `sumShift`. Generic indexes are not changed. Sums that contain `SEA`-ordered fermion operators are left unchanged.

The shift is applied to the whole summand; apply `sumExpand` first to canonicalize each term separately.

Only automatically generated names are renamed. Sums that differ by the choice of the dummy name, such as $\sum_k n_k$ and $\sum_p n_p$, are not identified; use `sumAbstractIndex` and `sumNameIndex`.

Example.

```
snegfermionoperators[ck];
snegmomentumindexes[{k, q}, Ns];
sumMomentumCanonical[sum[nc[ck[CR, k + q, UP], ck[AN, k, UP]], {k}]]
```

sumNameIndex

`sumNameIndex[expr, idx, {i, j, ...}]` replaces the abstract indexes `idx[1]`, `idx[2]`, ... in *expr* by the index names *i*, *j*, ...

`sumNameIndex` is the inverse of `sumAbstractIndex`: `sumNameIndex[ sumAbstractIndex[expr, idx], idx, names]` gives sums with common dummy names, so that equivalent sums are combined.

The list of names must contain at least as many entries as the largest number of indexes of a sum; abstract indexes without a name are left unchanged.

The names should be declared with `snegfreeindexes` (or as lattice or momentum indexes).

Example.

```
snegfreeindexes[i, j, k, l];
snegrealfunctions[f, g];
s = sum[f[i] g[j], {i, j}] + sum[f[k] g[l], {k, l}]
```

sumShift

`sumShift[expr, k -> k + q]` replaces the summation variable k by $k+q$ in all sums over k in *expr*.
`sumShift[expr, k -> -k + q]` applies the reflection $k \to -k+q$.
The change of variables $\sum_k f(k) = \sum_k f(\pm k + q)$ is exact if $k \to \pm k + q$ is a bijection of the index set. This holds for declared lattice and momentum indexes (`sneglatticeindexes`, `snegmomentumindexes`): momenta are defined modulo reciprocal-lattice vectors and sites modulo the lattice period.
k must be a declared lattice or momentum index, the coefficient of k must be ± 1, and the shift q must not contain k and must contain only quantities of the same type (momenta for a momentum index; lattice indexes and lattice vectors for a lattice index). Otherwise a message is issued and *expr* is returned unchanged.
Sums whose summand contains fermion operators with `SEA` ordering are not shifted (message `sumShift::sea`). For a linearized band with unbounded momenta, the shift is not a symmetry of the Fermi sea; this is the origin of the anomalous (Schwinger) term in the commutator of density operators of the Tomonaga-Luttinger model.
Only sums over k are changed; a free k outside sums is not replaced.
`sumMomentumCanonical` chooses the shifts automatically.
`sumShift::index` k is not a declared lattice or momentum index
`sumShift::shift` the substitution is not of the form $\pm k + q$ with q of the same index type
`sumShift::sea` the sum contains `SEA`-ordered operators and is not shifted

Example.

```
snegfermionoperators[ck];
snegmomentumindexes[{k, q}, Ns];
sumShift[sum[number[ck[k + q]], {k}], k -> k - q]
```

sumSimplify

`sumSimplify[expr]` simplifies an expression with symbolic sums by `Simplify` with additional transformation functions for sums.
The transformation functions are `SimplifyKD`, combinations of `Expand`, `sumExpand`, `Simplify`, `sumCollect`, and `sumSimplifyKD`, and the individual rules of `sumExpand`, `sumCollect`, and `sumSimplifyKD`. `Simplify` keeps the form with the smallest complexity.
Unlike `sumFullSimplify`, `sumSimplify` uses `Simplify` instead of `FullSimplify`, and it does not shift momenta or rename dummy indexes (`sumMomentumCanonical`). It is faster but finds fewer simplifications.

Example.

```
snegfermionoperators[c];
snegfreeindexes[k, p];
snegrealfunctions[ε];
sumSimplify[komutator[sum[ε[k] number[c[k], UP], {k}], c[CR, p, UP]]]
```

sumSimplifyKD

`sumSimplifyKD[expr]` performs the sums over Kronecker deltas in *expr*, $\sum_i \delta_{ij} f_i = f_j$, splits sums into products of independent sums, and performs lattice sums over Bloch phases.

For any declared indexes, a delta whose argument is a summation index i is eliminated: $\sum_i (\delta_{i,x} a_i + b_i) = a_x + \sum_i b_i$, where x does not contain i. Deltas inside the scalar marker `scalar[...]` are handled in the same way.

Sums that factorize are split, $\sum_{ij} a_i b_j = (\sum_i a_i)(\sum_j b_j)$, and summands of the form $a_i + b_j$ give $\sum_j 1 \sum_i a_i + \sum_i 1 \sum_j b_j$.

For lattice and momentum indexes (`sneglatticeindexes`, `snegmomentumindexes`), on which a shift $k \to \pm k + q$ is a bijection, `sumSimplifyKD` additionally:

solves deltas of linear combinations with coefficients ± 1: $\sum_k \delta_{p,k+q} f(k) = f(p-q)$, $\sum_k \delta_{p,q-k} f(k) = f(q-p)$; if several summation indexes can be eliminated, the last one in the index list is used;

performs the lattice sums $\sum_R e^{iK\cdot R} = N_s \delta_{K,0}$ and $\sum_k e^{ik\cdot X} = N_s \delta_{X,0}$ over Bloch phases `blochphase[k, R]` that enter the summand multiplicatively;

reduces deltas whose arguments differ by a combination of lattice quantities, also outside sums: $\delta_{k,k+q} = \delta_{q,0}$, $\delta_{R,R+1} = 0$.

Deltas are not solved if the coefficient of the summation index is not ± 1 ($\delta_{2k,q}$ has several solutions on the torus), if lattice and momentum quantities are mixed, or, for generic indexes, if the summation index occurs in a linear combination: on an open chain, $\sum_i \delta_{j,i+1} f(i) = f(j-1)$ fails at the boundary.

The rules are applied until the expression no longer changes (at most 50 iterations). The summands are not expanded. Deltas in products such as $\epsilon_k(\delta_{kp} c_k^\dagger - \ldots)$ are eliminated only after `sumExpand`. `sumCollectSimplifyKD` collects the deltas in each summand first.

Expressions without `KroneckerDelta` and `blochphase` are returned unchanged; in particular, sums are then not split into products.

`SimplifyKD` simplifies products and powers of deltas and step functions outside sums.

Example.

```
snegfreeindexes[i, j];
snegrealfunctions[f];
```

```
sumSimplifyKD[sum[KroneckerDelta[i, j] f[i], {i}]]
```

sumThread

`sumThread[expr]` threads the sums with a list as summand over the list: `sum[{a, b, ...}, it]` becomes `{sum[a, it], sum[b, it], ...}`.
`sum` does not thread over lists automatically; `sumThread` applies the rule `sum[{a, b, ...}, it] -> {sum[a, it], sum[b, it], ...}` repeatedly to *expr*.
`Thread[sum[{a, b}, {k}]]` threads over all list arguments, including the index list, and fails for lists of different lengths; use `sumThread`.

Example.

```
snegfermionoperators[c];
snegfreeindexes[k];
sum[{number[c[k], UP], number[c[k], DO]}, {k}]
```

15.6 States, Bases, and Matrices

ap

`ap[op, v]` applies the operator *op* to the state v in the occupation number representation.
`ap[op1, op2, ..., v]` applies the product $\hat{O}_1\hat{O}_2\cdots$ to v.
`ap[op, VACUUM]` applies *op* to `vacuum[]`.
v is a `vc` vector or a linear combination of `vc` vectors. *op* can be a single fermion operator, a product `nc[...]`, a sum of terms with scalar coefficients, or a ket-bra operator acting on the kets in `vc[..., ket[...]]`.
For the k-th operator a_k of `BASIS` and its adjoint, the rules are

$$a_k^\dagger|\ldots n_k \ldots\rangle = (-1)^{\sum_{l<k} n_l}\delta_{n_k,0}|\ldots 1 \ldots\rangle, \quad a_k|\ldots n_k \ldots\rangle = (-1)^{\sum_{l<k} n_l}\delta_{n_k,1}|\ldots 0 \ldots\rangle,$$

in agreement with the convention $\mathrm{vc}[n_1,\ldots] = (a_1^\dagger)^{n_1}(a_2^\dagger)^{n_2}\cdots|0\rangle$ (see `vc`). The implementation does not depend on the spin of the fermions.
Products act from the right: `ap[nc[A, B], v]` is `ap[A, ap[B, v]]`.
`ap` is linear in all arguments: sums are distributed and scalar factors are pulled out. Scalars are numbers and declared constants (`snegrealconstants`, `snegcomplexconstants`); undeclared symbols are treated as operators (see Possible Issues).
If a product `nc[..., v]` ends with a `vc` vector, `ap` is called automatically: `nc[op, v]` is `ap[op, v]`.
Kets: `ap[ket[k], v]` appends the ket to v, `vc[..., ket[k]]`; kets with complementary `Null` slots are combined (`ket[a, Null]` and `ket[Null, b]` give

`ket[a, b]`). A bra acting on the ket of v gives `braketrule[bra[...], ket[...]]`. Ket-bra operators such as `nc[ket[i], bra[j]]` thus act on the kets.
`ap[0, v]` and `ap[op, 0]` are 0; `ap[v]` is v.
`ap` is `Listable`.
An operator whose level is not in `BASIS` is not applied: `ap` returns unevaluated. Products that contain such operators lead to infinite recursion; the evaluation is aborted with `$RecursionLimit`. Always define the basis for all levels that occur in the operators.
`ap` is applicable to fermion operators and to bra-ket operators. Spin operators `S[...]` and boson operators are not applied; use kets for such degrees of freedom.

Example.

```
snegfermionoperators[c];
makebasis[{c[1], c[2]}];
vac = vacuum[];
ap[c[CR, 1, UP], vac] // InputForm
```

appendquantumnumber

`appendquantumnumber[bz, qn]` appends the quantum number *qn* to the quantum numbers of every subspace of the basis *bz*.
The result is `{{Append[qns, qn], states}, ...}`; the states are not changed.
`transformtoLR` uses `appendquantumnumber` to label the even and odd subspaces with the mirror parity $P = 1$ and $P = -1$ before joining them.

Example.

```
snegfermionoperators[d];
appendquantumnumber[qszbasis[{d[]}], 1]
```

applybasis

`applybasis[bz, f]` applies the function *f* to every state of the basis *bz* and keeps the quantum numbers.
bz must be a basis of the form `{{{qn...}, {states}}, ...}`, as tested by `bzQ`; otherwise `applybasis` returns unevaluated.
The result is `{{qn, f /@ states}, ...}`. The states are neither orthogonalized nor normalized, and states that become 0 are kept; use `transformbasis` to apply an operator followed by orthogonalization.

Example.

```
applybasis[{{{0}, {a, b}}, {{1}, {c}}}, f]
```

basis

`basis[c[i]]` gives the basis of the Fock space of the level `c[i]` as a list of creation operator strings.
The states are represented by operator strings acting on the vacuum: 1 (empty level), $c^\dagger_{i\uparrow}$, $c^\dagger_{i\downarrow}$, and $c^\dagger_{i\downarrow}c^\dagger_{i\uparrow}$ for spin-1/2 fermions (the last string in normal order, $c^\dagger_\downarrow c^\dagger_\uparrow|0\rangle = -c^\dagger_\uparrow c^\dagger_\downarrow|0\rangle$).
For fermions with spin s the list contains all 2^{2s+1} states, ordered by the number of particles.
The level is given without the creation/annihilation and spin indexes. `basis` is `Listable`.
`basis` does not define `BASIS` and does not depend on it. Bases of several levels are obtained with `direct`, or with the generators `nonebasis`, `qbasis`, `qszbasis`, ... for symmetry-adapted bases.

Example.

```
snegfermionoperators[d];
basis[d[]]
```

BASIS

`BASIS` is the current ordered list of creation operators used to interpret occupation vectors.
`makebasis` expands the levels into individual spin-orbitals. Position j in `vc` belongs to entry j of `BASIS`. A set bit specifies a creation operator; the order of its predecessors determines the sign on applying an annihilator.
Basis generators may change this global list. Save it with numerical matrices and with occupation vectors. `basis[c[i]]` instead constructs the many-body states of a single level.

Example.

```
snegfermionoperators[c];
makebasis[{c[2], c[1]}];
InputForm[BASIS]
```

basistensorproduct

`basistensorproduct[bz1, bz2, f]` gives the tensor product of the bases *bz1* and *bz2*: the products `nc[s1, s2]` of all states, with the quantum numbers `f[qn1, qn2]`; subspaces with equal quantum numbers are merged.
`basistensorproduct[bz1, bz2]` uses `f = List`.

bz1 and *bz2* are bases `{{qn, states}, ...}` in the creation operator representation (operator strings and kets). The state of the first basis stands to the left in the product.
f combines the quantum number lists of the two subspaces. For additive quantum numbers such as (Q, S_z), `f = Plus` adds them componentwise. The default `List` gives the nested quantum numbers `{qn1, qn2}`.
The result is merged and sorted with `mergebasis`. `directproductbasis` is the same construction without merging.
Product bases are used to add non-fermionic degrees of freedom (spin kets, phonon kets) to fermionic bases, and to build bases of large systems from bases of subsystems. For SU(2) quantum numbers, the product of highest-weight states is not a highest-weight basis of the total spin; use `SU2basistensorproduct`, which couples the multiplets with Clebsch-Gordan coefficients.
`basistensorproduct` does not call `makebasis`. Before converting the result to the occupation number representation, define `BASIS` for all levels.

Example. The (Q, S_z) basis of two levels from the bases of single levels:

```
snegfermionoperators[c];
bz = basistensorproduct[qszbasis[{c[1]}], qszbasis[{c[2]}], Plus]
```

bra

`bra[i]` represents the Dirac bra $\langle i|$.
`bra[i, j, ...]` represents the bra $\langle i, j, \ldots|$ of a product state; entries `Null` mark subsystems that the bra does not refer to.
`bra[i]` is the Hermitian conjugate of `ket[i]`: `conj[ket[i]]` is `bra[i]`.
In `nc` products, `nc[bra[i], ket[j]]` is evaluated by `braketrule[bra[i], ket[j]]` (by default δ_{ij}) if the bra and the ket refer to the same subsystems (the same pattern of `Null` entries). Bras of complementary subsystems are combined: `nc[bra[a, Null], bra[Null, b]]` is `bra[a, b]`.
A bra acting on a state `vc[..., ket[j]]` in the occupation number representation gives the braket with the ket of the state; see `ap`.
All other rules, examples, and applications are described on the page of `ket`.

Example.

```
{nc[bra[1], ket[1]], nc[bra[1], ket[2]], nc[bra[a], ket[b]]}
```

braket

`braket[a, op, b]` gives the matrix element $\langle a|\hat{O}|b\rangle$ in the occupation number representation; it is an alias of `braketvc`.

`braket[a, op, b]` evaluates to `braketvc[a, op, b]`, i.e. `scalarproductvc[a, ap[op, b]]`.
`braket` is `Listable` in all arguments.
`examples/diagonalize.m` uses `braket` for sums of matrix elements over the eigenstates.

Example.

```
snegfermionoperators[c];
makebasis[{c[1], c[2]}];
braket[vc[0, 0, 1, 0], hop[c[1], c[2]], vc[1, 0, 0, 0]]
```

braketrule

`braketrule[bra[i], ket[j]]` gives the scalar product $\langle i|j\rangle$.
`braketrule` is called when a bra and a ket are juxtaposed in an `nc` product, when a bra is applied to a `vc` state with a ket (`ap`), and by `scalarproductvc` for `vc` states with kets.
The default rule gives the product of `KroneckerDelta[i, j]` over all pairs of entries that are not `Null`: the kets form an orthonormal basis. For example, `braketrule[bra[a, Null, b], ket[c, Null, d]]` is $\delta_{ac}\delta_{bd}$.
The default rule applies only if the bra and the ket have the same pattern of `Null` entries; otherwise `braketrule` remains unevaluated.
Other scalar products can be implemented by additional definitions of `braketrule`, for example for non-orthogonal states. Since the default rule has a general pattern with a condition, definitions added later are stored after it and are not used for bras and kets of the default form. New rules must be placed before the default rule, e.g. with `DownValues[braketrule] = Prepend[DownValues[braketrule], rule]` (see Possible Issues).

Example.

```
{braketrule[bra[1], ket[1]], braketrule[bra[1], ket[2]], braketrule[bra[
    x], ket[y]]}
```

braketvc

`braketvc[a, op, b]` gives the matrix element $\langle a|\hat{O}|b\rangle$ of the operator *op* between the states *a* and *b* in the occupation number representation.
`braketvc[a, op, b]` is `scalarproductvc[a, ap[op, b]]`. The bra state *a* is conjugated.
`braketvc` is `Listable` in all arguments. `braket` is an alias.

The matrices of an operator in a list of states are obtained with `matrixrepresentationvc`, and in all pairs of invariant subspaces of a basis with `makeallmatricesbzvc`.
For states in the creation operator representation, use `braketop`.

Example. Hopping amplitude between two one-electron states:

```
snegfermionoperators[c];
snegrealconstants[t];
makebasis[{c[1], c[2]}];
braketvc[vc[0, 0, 1, 0], -t hop[c[1], c[2]], vc[1, 0, 0, 0]]
```

bzop2bzvc

`bzop2bzvc[bz]` converts the basis *bz* from the creation operator representation to the occupation number representation.
`bzop2bzvc[bz, vac]` applies the operator expressions of the states to the vector *vac*. *bz* is a basis `{{qn, states}, ...}` (see `makematricesbzvc`). Each state *f* is converted to $f|\mathrm{vac}\rangle$ with `ap[f, vac]`. The default *vac* is `vacuum[]`, the empty state `vc[0, 0, ...]` of the current `BASIS`.
The basis generators `qszbasis`, `qsbasis`, ... set `BASIS` with `makebasis`. For bases constructed otherwise, call `makebasis` with all levels that occur before the conversion.
`bzvc2bzop` is the inverse conversion. For larger systems, convert operator-form bases with `bzop2bzvc` and compute matrices with `makematricesbzvc`, which is much faster than `makematricesbzop`.

Example.

```
snegfermionoperators[c];
bz = qsbasis[{c[1], c[2]}]
```

bzQ

`bzQ[bz]` gives `True` if *bz* has the form of a basis, a nonempty list of pairs `{{qn...}, {states...}}`, and `False` otherwise.
A basis is a list `{{qn1, states1}, {qn2, states2}, ...}` of invariant subspaces, where the quantum numbers `qn` and the states are lists; see `makematricesbzvc`.
`bzQ` checks only the structure, not the states. Lists of matrices `{{qn, mat}, ...}` as returned by `makematricesbzvc` also have this form.
The empty list `{}` gives `False`.
`makematricesbzvc`, `makematricesbzop`, `showbzmat`, and `applybasis` require arguments that satisfy `bzQ`.

Example.

```
snegfermionoperators[c];
bzQ[qszbasisvc[{c[1]}]]
```

bzvc2bzop

`bzvc2bzop[bz]` converts the basis *bz* from the occupation number representation to the creation operator representation.
bz is a basis `{{qn, states}, ...}` (see `makematricesbzvc`). Each state is converted with `vc2ops`: `vc[n1, n2, ...]` becomes the product of the creation operators of the occupied states in the order of the global list `BASIS`, $(c_1^\dagger)^{n_1}(c_2^\dagger)^{n_2}\cdots$, with the vacuum implied, and `vc[0, 0, ...]` becomes `1`. The quantum numbers are unchanged.
Kets at the end of a vector, `vc[n1, ..., ket[k]]`, become `nc[..., ket[k]]`.
The conversion uses the current `BASIS`; vectors of a different length give the message `vc2ops::incom`.
`bzop2bzvc` is the inverse conversion. Both representations give the same matrices: `makematricesbzop[op, bzvc2bzop[bz]]` equals `makematricesbzvc[op, bz]`.
The creation operator representation is convenient for symbolic manipulation, e.g. substitution of operators in `transformtoLR`, and for display. The occupation number representation is faster for matrix calculations.

Example. The (Q, S_z) basis of a single level in both representations:

```
snegfermionoperators[d];
bz = qszbasisvc[{d[]}]
```

decomposeop

`decomposeop[v, basis]` gives the components $\langle b_j|v\rangle$ of the state v with respect to the states b_j of *basis*, all in the creation operator representation.
The result is the list `scalarproductop[#, v] & /@ basis`, computed with vacuum expectation values. See `decomposevc` for the occupation number representation.

Example.

```
snegfermionoperators[d];
decomposeop[(d[CR, UP] + I d[CR, DO])/Sqrt[2], basis[d[]]]
```

decomposevc

`decomposevc[v, basis]` gives the components $\langle b_j|v\rangle$ of the state v with respect to the states b_j of *basis*, in the occupation number representation.
The result is the list `scalarproductvc[#, v] & /@ basis`. For an orthonormal *basis* spanning v, $v = \sum_j \langle b_j|v\rangle\, b_j$, i.e. `v == decomposevc[v, basis] . basis`.
`decomposeop` is the corresponding function for the creation operator representation.

Example.

```
snegfermionoperators[d];
makebasis[{d[]}];
decomposevc[(vc[1, 0] + I vc[0, 1])/Sqrt[2], {vc[0, 0], vc[1, 0], vc[0,
    1], vc[1, 1]}]
```

direct

`direct[l1, l2]` gives the flattened list of all products `nc[a, b]` with *a* from the list *l1* and *b* from *l2*.
`direct[l1, l2]` is `Flatten[outer[l1, l2]]`, where `outer` is `Outer` with `nc` as the product. The elements of *l1* stand to the left.
`direct` is used to build bases of several levels from single-level bases (`basis`) in the creation operator representation.

Example.

```
direct[{2, 3}, {a, b}]
```

directproductbasis

`directproductbasis[bz1, bz2, f]` gives the product basis of *bz1* and *bz2*: for every pair of subspaces, the products `nc[s1, s2]` of their states, with the quantum numbers `f[qn1, qn2]`.
`directproductbasis[bz1, bz2]` uses `f = Join`.
Unlike `basistensorproduct`, the subspaces are not merged: the result has one element for each pair of subspaces of *bz1* and *bz2*, in the order of `Outer`. Use `mergebasis` to merge subspaces with equal quantum numbers.
The default `Join` concatenates the quantum number lists: `{Q1, Sz1}` and `{Q2, Sz2}` give `{Q1, Sz1, Q2, Sz2}`.
`quickDBL` and `quickDBLSZ` are built on `directproductbasis`.
`directproductbasis` does not call `makebasis`.

Example.

```
snegfermionoperators[c];
directproductbasis[qbasis[{c[1]}], qbasis[{c[2]}]]
```

dropemptysubspaces

`dropemptysubspaces[bz]` removes the subspaces without states from the basis *bz*. *bz* is a list `{{qn, states}, ...}`; elements with `states === {}` are removed. The order of the remaining subspaces is preserved.

Projections and symmetry transformations can leave subspaces empty. `transformbasis`, `transformtoLR`, `transformtoLRvc`, and `transformQStoIS` apply `dropemptysubspaces` to their results.

Example.

```
dropemptysubspaces[{{{0}, {}}, {{1}, {a, b}}, {{2}, {}}}]
```

expvvc

`expvvc[op, a]` gives the diagonal matrix element $\langle a|\hat{O}|a\rangle$ of the operator *op* in the state *a* in the occupation number representation.

`expvvc[op, a]` is `braketvc[a, op, a]`, i.e. `scalarproductvc[a, ap[op, a]]`.

The result is not divided by $\langle a|a\rangle$: for a normalized expectation value, the state must be normalized.

`expvvc` is `Listable` in both arguments. The corresponding function for the creation operator representation is `expvop`.

Example.

```
snegfermionoperators[c];
makebasis[{c[1], c[2]}];
singlet = (vc[1, 0, 0, 1] - vc[0, 1, 1, 0])/Sqrt[2];
expvvc[spinspin[c[1], c[2]], singlet]
```

fixspin

`fixspin[v, s, sz]` gives the state with the spin projection *sz* of the multiplet with total spin *s* whose highest-weight state ($S_z = s$) is *v*.

v is a state in the creation-operator representation with $S_z = s$. `fixspin` applies `spindown` $s - sz$ times; each application is normalized, so the result is the normalized state $|s, sz\rangle$ with the standard phases.

`fixspin[v, s, s]` is *v*. For $sz > s$ the expression remains unevaluated.

`fixspin` and `fixspinket` are used as arguments of `SU2basistensorproduct`, which builds bases with good total spin from Clebsch-Gordan combinations of multiplets; NRG Ljubljana uses them in this way.

Example. The $S_z = 0$ and $S_z = -1$ triplet states of two electrons:

```
snegfermionoperators[c];
up2 = nc[c[CR, 1, UP], c[CR, 2, UP]];
{fixspin[up2, 1, 0], fixspin[up2, 1, -1]}
```

goodopQ

`goodopQ[c[t, i, σ]]` gives `True` if the operator $c_{i\sigma}$ belongs to the single-particle basis `BASIS` defined by `makebasis`, and `False` otherwise.
The creation/annihilation index `t` is disregarded.
`ap` applies an operator to a `vc` vector only if `goodopQ` gives `True`; `op2ndx` returns a position only for such operators.
`goodopQ` remains unevaluated for expressions without indexes, such as the level `c[]`.

Example.

```
snegfermionoperators[c, d];
makebasis[{c[1]}];
{goodopQ[c[CR, 1, UP]], goodopQ[c[AN, 1, DO]], goodopQ[c[CR, 2, UP]],
    goodopQ[d[CR, UP]]}
```

interleave

`interleave[{a1, a2, ...}, {b1, b2, ...}, ...]` gives `{a1, b1, ..., a2, b2, ..., ...}`.
`interleave[l1, l2, ...]` is `Flatten[Transpose[{l1, l2, ...}], 1]`. The lists must have equal lengths.
In the occupation number vectors `vc[...]`, the spin-up and spin-down states of each level are adjacent: `vc[n1↑, n1↓, n2↑, n2↓, ...]`. `qszbasisvc` uses `interleave` to combine the spin-up and spin-down occupations.

Example.

```
interleave[{a1, a2, a3}, {b1, b2, b3}]
```

ket

`ket[i]` represents the Dirac ket $|i\rangle$.
`ket[i, j, ...]` represents the product state $|i, j, ...\rangle$ of several subsystems.
`ket[]` is the ket of a system without quantum numbers.
Kets and bras describe degrees of freedom that are not represented by fermion operators: impurity spins, phonons, orbital pseudospins, or abstract states.
Operators on these degrees of freedom are written as linear combinations of ket-bra products `nc[ket[i], bra[j]]` $= |i\rangle\langle j|$.
The labels can be numbers or symbols. A label `Null` marks a subsystem that the ket does not refer to: `ket[a, Null]` is a state of the first subsystem only. With `PrettyOutput`, `Null` is displayed as a small circle.
Rules in `nc` products:
`nc[bra[i], ket[j]]` $\langle i|j\rangle =$ `braketrule[bra[i], ket[j]]`, by default δ_{ij}

`nc[ket[a, Null], ket[Null, b]]` `ket[a, b]`: kets of complementary subsystems are combined

`nc[ket[a, Null], bra[c, Null], ket[Null, b], bra[Null, d]]` `nc[ket[a, b], bra[c, d]]`

`nc[ket[i], c[CR, ...]]` `nc[c[CR, ...], ket[i]]`: kets and bras are moved to the right of fermion operators

A bra and a ket are evaluated only if they have the same pattern of `Null` entries (the same subsystems). Kets are combined only if their non-`Null` entries are at different positions; otherwise the product stays unevaluated.

`conj[ket[i]]` is `bra[i]`, and `conj[bra[i]]` is `ket[i]`; `conj` reverses products and conjugates the coefficients.

In the occupation number representation, a ket is appended to the `vc` vector, `vc[n1, ..., ket[i]]`; see `ap` and `vc`.

Kets commute with fermion operators. For fermionic degrees of freedom, use fermion operators instead.

Example.

```
{ket[1], bra[1], ket[x, y]}
```

ketbratensorproduct

`ketbratensorproduct[x1, x2]` gives the tensor product of the expressions *x1* and *x2* built from kets and bras, $X_1 \otimes X_2$.

`ketbratensorproduct[x1, x2, x3, ...]` gives the tensor product of several factors.

`ketbratensorproduct[x]` is x.

The kets and bras of *x1* are padded on the right with `Null` entries for the slots of *x2*, and those of *x2* on the left with `Null` entries for the slots of *x1*; the two expressions are then multiplied with `nc`, which combines kets and bras with complementary `Null` patterns. The number of slots of each factor is the maximal number of entries of its kets and bras.

Typical uses are operators on composite systems built from operators on the factors, such as $S^z \otimes 1$, or the operators of one phonon mode in a multi-mode phonon space (`phononnumber[i, {N1, N2, ...}]` is implemented this way).

Each factor should be an operator (sums of ket-bra products) or a state on its own subsystem; the identity of a subsystem must be written explicitly (e.g. `spinketbraI[s]`).

Example.

```
ketbratensorproduct[ket[1], ket[2]]
```

keymerge

`keymerge[list]` merges the elements `{key, values}` of *list* with equal keys into a single element `{key, allvalues}`.
The keys are sorted (`Union`), and for each key the values of all elements with this key are joined and flattened with `Flatten`.
Applied to a list of subspaces `{{qn, states}, ...}`, `keymerge` combines the states of subspaces with equal quantum numbers. It is used in the construction of the (Q, S) bases. `mergebasis` performs the same operation.
The values are flattened completely, so they must not be lists of lists.

Example.

```
keymerge[{{1, {a, b}}, {2, {c}}, {1, {d}}}]
```

keyselect

`keyselect[list, key]` gives the elements of *list* whose first part equals *key*.
`keyselect[list, key]` is `Select[list, First[#] == key &]`. Applied to a basis or to a list of matrices `{{qn, ...}, ...}`, it gives the list of elements for the subspace with quantum numbers *key*, i.e. `{{key, states}}`; use `keyselect[bz, key][[1, 2]]` to extract the states or the matrix.
The comparison uses `Equal`. Patterns are not supported as keys; use `Cases[list, {{0, _}, _}]` to select by pattern.

Example. The half-filled singlet subspace of the two-site (Q, S) basis:

```
snegfermionoperators[c];
bz = qsbasisvc[{c[1], c[2]}];
keyselect[bz, {0, 0}]
```

lrmap

`lrmap[{a, b, ..., z}]` gives the substitution rules for the creation operators of the mirror reflection that maps the list of levels onto its reverse, $a \leftrightarrow z$, $b \leftrightarrow y$, ...
The levels are given without the creation and spin indexes, e.g. `c[1]` or `d[]`. Each level is expanded into its creation operators, `c[CR, 1, UP]` and `c[CR, 1, DO]` for spin 1/2 and all spin components for other spins (as in `makebasis`). The rules map the creation operators of the k-th element to those of the k-th element from the end, with the same spin.
The middle element of a list of odd length is mapped onto itself.
Elements can be lists of levels. A sublist is mapped onto the sublist at the mirror position, element by element; the levels inside a sublist are not reversed. This describes, e.g., several orbitals or a chain of sites on each side.

Only creation operators appear in the rules, which is sufficient for basis states in the creation operator representation. `transformtoLR` and `transformtoLRvc` apply the rules to the basis states.
The reflection maps the vacuum onto itself and acts on a state $f(c^\dagger)|0\rangle$ by substitution, $Pf(c_i^\dagger)|0\rangle = f(c_{\bar{i}}^\dagger)|0\rangle$, where $\bar{i}$ is the mirror image of i; this fixes the sign convention of the parity eigenvalues.

Example. Two sites exchanged by the reflection:

```
snegfermionoperators[c];
lrmap[{c[1], c[2]}]
```

makeallmatricesbzop

`makeallmatricesbzop[op, bz]` gives the list `{{{qn1, qn2}, mat}, ...}` of the nonzero blocks of the operator *op* between all pairs of invariant subspaces of the basis *bz*, with the basis states in the creation operator representation.
`makeallmatricesbzop` is the variant of `makeallmatricesbzvc` for bases whose states are operator expressions (`qsbasis`, `qszbasis`, `transformtoLR`, `bzvc2bzop`). The block for the pair `{qn1, qn2}` is $M_{ij} = \langle i|\hat{O}|j\rangle$ with $|i\rangle$ in `qn1` and $|j\rangle$ in `qn2`, computed with `matrixrepresentationop`.
Vanishing blocks are omitted. See `makeallmatricesbzvc` for details and caveats, in particular for (Q, S) bases.
The creation operator representation is slow; for larger bases use `bzop2bzvc` and `makeallmatricesbzvc`.

Example. Matrix elements of $c_{1\downarrow}^\dagger$ between the (Q, S) subspaces of the Hubbard dimer:

```
snegfermionoperators[c];
makeallmatricesbzop[c[CR, 1, DO], qsbasis[{c[1], c[2]}]]
```

makeallmatricesbzvc

`makeallmatricesbzvc[op, bz]` gives the list `{{{qn1, qn2}, mat}, ...}` of the nonzero blocks `mat` of the operator *op* between the invariant subspaces `qn1` and `qn2` of the basis *bz*, with the basis states in the occupation number representation.
bz is a basis `{{qn, states}, ...}` as described in `makematricesbzvc`.
For every ordered pair of subspaces, the block is $M_{ij} = \langle i|\hat{O}|j\rangle$ with $|i\rangle$ in the subspace `qn1` (bra, final state) and $|j\rangle$ in the subspace `qn2` (ket, initial state). The block has the dimensions `Length[states1]` × `Length[states2]`.
Blocks with all elements zero are omitted; the remaining pairs are the transitions allowed by the selection rules of *op*. Blocks whose elements cannot be decided to vanish, e.g. with symbolic coefficients, are kept. The pairs are ordered as in `Outer[List, bz, bz, 1]`.

Unlike `makematricesbzvc`, the blocks are not simplified.
`makeallmatricesbzvc` is needed for operators that change the quantum numbers, such as creation and annihilation operators in the Lehmann representation of Green's functions,
$A(\omega) = \sum_n |\langle n|c^\dagger|0\rangle|^2 \delta(\omega - E_n + E_0) + \sum_n |\langle n|c|0\rangle|^2 \delta(\omega + E_n - E_0)$.
In a (Q, S) basis, only the states with $S_z = S$ are present, and the blocks contain only the matrix elements between these states. For non-scalar operators, the full matrix elements follow from the Wigner-Eckart theorem; otherwise, use a (Q, S_z) basis. The number of pairs grows quadratically with the number of subspaces; every pair is computed.

Example. Blocks of the creation operator $c^\dagger_\uparrow$ of a single level in the (Q, S_z) basis; $c^\dagger_\uparrow$ raises Q by 1 and S_z by 1/2:

```
snegfermionoperators[d];
bz = qszbasisvc[{d[]}];
makeallmatricesbzvc[d[CR, UP], bz]
```

makebasis

`makebasis[{c[i], d[j], ...}]` defines the single-particle basis for the occupation number representation from the levels `c[i]`, `d[j]`, ... and returns it.
`makebasis[c[i], d[j], ...]` is equivalent.
The levels are given without the creation/annihilation index and without the spin index: `c[i]` stands for the operators $c_{i\sigma}$, `c[]` is a level without further indexes. Every level is expanded into creation operators for all spin projections, from the highest to the lowest: `c[CR, i, UP]`, `c[CR, i, DO]` for spin-1/2 fermions, `c[CR, i, s]`, `c[CR, i, s-1]`, ..., `c[CR, i, -s]` for fermions declared with spin s, and `c[CR, i, 0]` for spinless fermions.
The result is stored in the global variable `BASIS`. The position of an operator in `BASIS` is the position of its occupation number in the `vc` vectors; see `op2ndx` and `ndx2op`. For levels $c_1, c_2, \ldots$ the order is
$c^\dagger_{1\uparrow}, c^\dagger_{1\downarrow}, c^\dagger_{2\uparrow}, c^\dagger_{2\downarrow}, \ldots$,
and $\mathrm{vc}[n_1, n_2, \ldots] = (c^\dagger_{1\uparrow})^{n_1} (c^\dagger_{1\downarrow})^{n_2} \cdots |0\rangle$.
The basis generators `qszbasisvc`, `qsbasisvc`, `qbasisvc`, `szbasisvc`, `sbasisvc`, `nonebasisvc`, and their operator-form versions call `makebasis` with their argument. Functions that build product bases from subsystems, such as `quickDBL`, call `makebasis` for the whole system at the end.
`BASIS` is global: there is one occupation number representation at a time. `vc` vectors, `vacuum[]`, and the results of `ap` and `vc2ops` refer to the current `BASIS`.
The levels can be nested in lists; the list is flattened. Duplicate levels are not removed.

Example. Single-particle basis of two spin-1/2 levels:

```
snegfermionoperators[c];
makebasis[{c[1], c[2]}]
```

makematricesbzop

`makematricesbzop[op, bz]` gives the list `{{qn, mat}, ...}` of the matrices of the operator *op* in the invariant subspaces of the basis *bz*, with the basis states in the creation operator representation.
`makematricesbzop` is the variant of `makematricesbzvc` for bases whose states are operator expressions: a state $f(c^\dagger)|0\rangle$ is written as the expression $f(c^\dagger)$, e.g. `nc[c[CR, 1, UP], c[CR, 2, DO]]`, and the vacuum as `1`. Such bases are produced by `qszbasis`, `qsbasis`, `transformtoLR`, and `bzvc2bzop`. See `makematricesbzvc` for the structure of a basis.
For each subspace with states $|i\rangle$, the matrix is $M_{ij} = \langle i|\hat{O}|j\rangle$, computed with `matrixrepresentationop` as vacuum expectation values $\langle 0|f_i^\dagger \hat{O} f_j|0\rangle$ (Wick's theorem). The matrices are simplified with `Simplify`.
The creation operator representation does not depend on the global `BASIS`, but it is much slower than the occupation number representation. For more than a few levels, convert the basis with `bzop2bzvc` and use `makematricesbzvc`.
The states may contain kets for additional degrees of freedom, e.g. phonons (`transformtoPH`) or spins (`spinket`); their scalar products are given by `braketrule`. Matrix elements between different subspaces are not computed; see `makeallmatricesbzop`.
`makematricesbzop` returns unevaluated if *bz* is not of the form tested by `bzQ`.

Example. Hubbard dimer in the (Q, S) basis in the creation operator representation:

```
snegfermionoperators[c];
snegrealconstants[t, U];
bz = qsbasis[{c[1], c[2]}]
```

makematricesbzvc

`makematricesbzvc[op, bz]` gives the list `{{qn, mat}, ...}` of the matrices `mat` of the operator *op* in the invariant subspaces of the basis *bz*, labeled by their quantum numbers `qn`; the basis states are in the occupation number representation.
A basis in SNEG is a list `{{qn1, states1}, {qn2, states2}, ...}`. Each element describes one invariant subspace (symmetry sector): `qn` is the list of its quantum numbers and `states` the list of its basis states. `bzQ` tests for this form.
The states are either occupation number vectors `vc[n1, n2, ...]` (functions with names ending in `vc`) or operator expressions, strings of creation operators acting on

the vacuum (functions ending in `op`, and the basis generators without `vc`). `bzvc2bzop` and `bzop2bzvc` convert between the two forms.

In `vc[n1, n2, ...]`, $n_k \in \{0, 1\}$ is the occupation of the k-th single-particle state of the global list `BASIS`, set by `makebasis`. For spin-1/2 fermion levels $c_1, c_2, \ldots$, the order is $c^\dagger_{1\uparrow}, c^\dagger_{1\downarrow}, c^\dagger_{2\uparrow}, \ldots$, and `vc[n1, n2, ...]` is the state $(c^\dagger_{1\uparrow})^{n_1}(c^\dagger_{1\downarrow})^{n_2}\cdots|0\rangle$.

Bases are generated by the following functions, which also call `makebasis`:

`qszbasisvc[{c[1], ...}]` $\{Q, S_z\}$ charge and spin projection

`qsbasisvc[{c[1], ...}]` $\{Q, S\}$ charge and total spin; only the states with $S_z = S$

`qbasisvc`, `szbasisvc`, `sbasisvc` $\{Q\}$, $\{S_z\}$, $\{S\}$ a single quantum number

`nonebasisvc {}` the whole Fock space as a single subspace

The charge is measured from half filling, $Q = N - N_{\text{levels}}$, where N is the number of electrons and N_{levels} the number of spin-1/2 levels. `transformQStoIS`, `transformtoLRvc`, `transformtoPH`, and `transformbasis` derive further bases with isospin, parity, phonons, or projections.

`makematricesbzvc[op, bz]` gives, for each subspace, the matrix $M_{ij} = \langle i|\hat{O}|j\rangle$, where $|i\rangle$ and $|j\rangle$ run over the states of the subspace, in the order of `bz`. The matrices are simplified with `Simplify`.

For an orthonormal basis and an operator that commutes with the symmetry generators (a Hamiltonian with the symmetry of the basis), the matrices are the blocks of the block-diagonal operator. The full spectrum is the union of the eigenvalues of all blocks; in a (Q, S) basis, every eigenvalue in the subspace (Q, S) is $(2S + 1)$-fold degenerate. The eigenvector $(v_1, v_2, \ldots)$ of a block is the state $\sum_i v_i|i\rangle$, obtained as `v . states`.

Matrix elements between different subspaces are not computed. For operators that change the quantum numbers, such as $c^\dagger_{i\sigma}$, use `makeallmatricesbzvc`.

`makematricesbzvc` returns unevaluated if *bz* is not of the form tested by `bzQ`.

The states `vc[...]` refer to the current value of `BASIS`. Calling a basis generator for another set of levels redefines `BASIS`, and the old states are then no longer valid: operators of levels that are not in `BASIS` are not applied (see `matrixrepresentationvc`), and products of such operators lead to infinite recursion. Generate the basis again with `makebasis` or the basis generator before computing matrices.

Example. Basis of the Hubbard dimer with well-defined charge Q and total spin S:

```
snegfermionoperators[c];
snegrealconstants[t, U];
bz = qsbasisvc[{c[1], c[2]}]
```

mapbasis

`mapbasis[bz1, bz2, f]` gives the basis with the quantum numbers of *bz1* and the states `f[states1, states2]` of corresponding subspaces of *bz1* and *bz2*.
`mapbasis[bz1, bz2]` uses `f = (states1 + states2)/2`.
bz1 and *bz2* must have identical lists of quantum numbers, in the same order. Otherwise, the message `mapbasis::compat` is issued and `Null` is returned.
f receives the two lists of states of a subspace and must return a list of states.
`mapbasis` is used to construct parity-adapted bases in `transformtoLR`: with the mirror-transformed basis $P\,bz$, the combinations $(bz + P\,bz)/2$ and $(bz - P\,bz)/2$ contain the even and odd states, which are then orthonormalized with `orthogbasisop`.

Example.

```
mapbasis[{{{0}, {a, b}}, {{1}, {c}}}, {{{0}, {x, y}}, {{1}, {z}}}]
```

matrixrepresentationop

`matrixrepresentationop[op, {f1, f2, ...}]` gives the matrix $M_{ij} = \langle 0|f_i^\dagger \hat{O} f_j|0\rangle$ of the operator *op* between the states $f_i|0\rangle$ given in the creation operator representation.
`matrixrepresentationop[op, {g1, g2, ...}, {f1, f2, ...}]` gives the rectangular matrix $M_{ij} = \langle 0|g_i^\dagger \hat{O} f_j|0\rangle$.
In the creation operator representation, a state is an operator expression *f* that creates it from the vacuum: `nc[c[CR, 1, UP], c[CR, 2, DO]]` is $c^\dagger_{1\uparrow} c^\dagger_{2\downarrow}|0\rangle$, and `1` is the vacuum. The vacuum is implied. States written with a trailing `VACUUM` also work, but the two forms must not be mixed.
The matrix elements are computed with `scalarproductop[fi, nc[op, fj]]`, i.e. as vacuum expectation values with `vevwick`. Scalar coefficients of the bras are complex conjugated.
The states may also be (or end with) kets, `ket[...]`, for spins, phonons, or other degrees of freedom; operators on them are written with `ket` and `bra`, e.g. `spinketbraP`, `phononplus`. Their overlaps are given by `braketrule`.
Unlike `matrixrepresentationvc`, no global `BASIS` is needed. The calculation is much slower than in the occupation number representation; convert large bases with `bzop2bzvc`.
The states do not need to be orthonormal; the result is then the matrix of matrix elements.

Example. Hopping between two sites in the basis of one-electron states with spin up:

```
snegfermionoperators[c];
snegrealconstants[t];
matrixrepresentationop[-t hop[c[1], c[2]], {c[CR, 1, UP], c[CR, 2, UP]}]
```

matrixrepresentationvc

`matrixrepresentationvc[op, {v1, v2, ...}]` gives the matrix $M_{ij} = \langle v_i|\hat{O}|v_j\rangle$ of the operator *op* between the states v_i in the occupation number representation.
`matrixrepresentationvc[op, {u1, u2, ...}, {v1, v2, ...}]` gives the rectangular matrix $M_{ij} = \langle u_i|\hat{O}|v_j\rangle$.
The states are linear combinations of occupation number vectors `vc[n1, n2, ...]`, where n_k is the occupation of the k-th single-particle state of the global list `BASIS` (set by `makebasis` or by the basis generators such as `qszbasisvc`). The vector `vc[n1, n2, ...]` is $(c_1^\dagger)^{n_1}(c_2^\dagger)^{n_2}\cdots|0\rangle$, with the creation operators in the order of `BASIS`; the fermionic signs follow from this order.
The operator is applied with `ap`, and the matrix elements are the scalar products `scalarproductvc[ui, ap[op, vj]]`, which are antilinear in the first argument. The result is `Outer[scalarproductvc, bras, ap[op, kets]]`.
The states do not need to be normalized or orthogonal. The result is then the matrix of matrix elements, not the matrix of the operator in a non-orthonormal basis; for the identity operator, it is the Gram matrix $\langle v_i|v_j\rangle$.
The rows correspond to the bras and the columns to the kets. With `matrixrepresentationvc[op, states]`, the matrix applied to coefficient vectors gives the action of $\hat{O}$ in the basis: if $|\psi\rangle = \sum_j a_j|v_j\rangle$ and the basis is orthonormal, $\hat{O}|\psi\rangle = \sum_i (Ma)_i|v_i\rangle$.
The states may end with kets of auxiliary degrees of freedom (`ket[...]`, e.g. phonons or spins); their overlaps are given by `braketrule`.
The result is not simplified. Operators of levels that are not in `BASIS` are not applied, and the result stays unevaluated.
`matrixrepresentationvcfast` and `matrixrepresentationvcsparse` give the same matrices faster for bases of single occupation number vectors.

Example. Matrix of the hopping operator $c_{1\uparrow}^\dagger c_{2\uparrow}$ + h.c. between the one-electron states with spin up:

```
snegfermionoperators[c];
makebasis[{c[1], c[2]}];
BASIS
```

matrixrepresentationvcfast

`matrixrepresentationvcfast[op, {v1, v2, ...}]` gives the matrix $\langle v_i|\hat{O}|v_j\rangle$ like `matrixrepresentationvc`, using a faster algorithm.
`matrixrepresentationvcfast[op, {u1, ...}, {v1, ...}]` gives the rectangular matrix $\langle u_i|\hat{O}|v_j\rangle$.

The result is the same as that of `matrixrepresentationvc`; see that page for the conventions.
The fast algorithm applies the operator once to each ket, $\hat{O}|v_j\rangle = \sum_k a_{kj}|w_k\rangle$, and reads off the coefficients of the bra vectors, instead of computing all scalar products. It is used when every bra is a single occupation number vector with a coefficient, `x vc[...]`, the bras are distinct, and they contain no kets. Otherwise `matrixrepresentationvc` is called.
For the Fock space of four spin-1/2 levels (256 states), the Hubbard Hamiltonian takes about a third of the time of `matrixrepresentationvc`.
The result is not simplified.

Example. Hamiltonian of the Hubbard dimer in the Fock space of 16 states:

```
snegfermionoperators[c];
snegrealconstants[t, U];
fock = nonebasisvc[{c[1], c[2]}][[1, 2]];
H = -t hop[c[1], c[2]] + U Total[hubbard[{c[1], c[2]}]];
m = matrixrepresentationvcfast[H, fock];
Dimensions[m]
```

matrixrepresentationvcsparse

`matrixrepresentationvcsparse[op, {v1, v2, ...}]` gives the matrix $\langle v_i|\hat{O}|v_j\rangle$ of the operator *op* as a `SparseArray`.
The matrix elements are those of `matrixrepresentationvc[op, states]`; see that page for the conventions.
For bases of distinct single occupation number vectors with coefficients, `x vc[...]`, without kets, the nonzero elements are collected directly from $\hat{O}|v_j\rangle$. Otherwise the dense matrix of `matrixrepresentationvc` is converted to a `SparseArray`.
Hamiltonians of lattice models with local interactions are sparse in the occupation number basis. Sparse matrices can be diagonalized partially with `Eigenvalues[m, k]` and `Method -> "Arnoldi"`.
Only the square form with a single list of states is defined; `matrixrepresentationvcsparse[op, bras, kets]` stays unevaluated.

Example. Hamiltonian of the Hubbard chain with three sites in the Fock space of 64 states, for $t = 1$, $U = 4$:

```
snegfermionoperators[c];
fock = nonebasisvc[{c[1], c[2], c[3]}][[1, 2]];
H = -hop[c[1], c[2]] - hop[c[2], c[3]] + 4 Total[hubbard[{c[1], c[2], c
    [3]}]];
m = matrixrepresentationvcsparse[H, fock]
```

mergebasis

`mergebasis[{{qn1, states1}, {qn2, states2}, ...}]` combines the states of all elements with equal quantum numbers into one subspace.
The result is a list `{{qn, states}, ...}` with one element for each distinct quantum number `qn`, sorted in canonical order. The states of elements with the same `qn` are joined in the order of their appearance.
The quantum numbers are compared with `SameQ`; they are usually lists `{Q, Sz}`, ..., but can be any expressions.
Typical uses: removing a quantum number from a basis (e.g. obtaining the Q basis from the (Q, S_z) basis), replacing quantum numbers by functions of them (e.g. charge parity $(-1)^Q$), and combining the products of subsystem bases with additive quantum numbers.
`keymerge` is an equivalent function.

Example.

```
mergebasis[{{{0}, {a1}}, {{1}, {b1}}, {{0}, {a2, a3}}}]
```

ndx2op

`ndx2op[n]` gives the creation operator at position n of the single-particle basis `BASIS` defined by `makebasis`.
`ndx2op[n]` is `BASIS[[n]]`. It always returns a creation operator.
`ndx2op` is `Listable`. The inverse function is `op2ndx`.
For n larger than the length of `BASIS`, the `Part::partw` message is issued.

Example.

```
snegfermionoperators[c, d];
makebasis[{c[], d[]}];
ndx2op[3]
```

nnop

`nnop[level]` specifies the sublattice phase exponent of *level* for the quick isospin basis constructors.
Assign integer values before building the basis. Only the parity matters: the coefficient of the local pair operator is $(-1)^{n_i}$. For real hopping between opposite sublattices choose opposite signs. Equal signs still generate a charge SU(2) algebra but generally not a symmetry of that hopping.
`nnop` is a model definition. It does not infer a sublattice assignment from the Hamiltonian. Give a value for every participating level, including a local impurity.

Example.

```
snegfermionoperators[c];
nnop[c[1]]=0; nnop[c[2]]=1;
tm = Total[isospinminus[#,nnop[#]] & /@ {c[1],c[2]}];
Expand[komutator[hop[c[1],c[2]],tm]]
```

nonebasis

`nonebasis[{c[1], c[2], ...}]` gives the basis of the full Fock space of the levels `c[1]`, `c[2]`, ... in the creation operator representation, as a single subspace without quantum numbers.
`nonebasis[ops]` is `bzvc2bzop[nonebasisvc[ops]]`. The result is `{{{}, {states}}}`.
`nonebasis` calls `makebasis` with its argument.
NRG Ljubljana uses `nonebasis` for the symmetry type `NONE`.

Example.

```
snegfermionoperators[d];
nonebasis[{d[]}]
```

nonebasisvc

`nonebasisvc[{c[1], c[2], ...}]` gives the basis of the full Fock space of the levels `c[1]`, `c[2]`, ... in the occupation number representation, as a single subspace without quantum numbers.
The result is `{{{}, {states}}}` with all 2^{N_s} occupation number states in lexicographic order, where N_s is the length of `BASIS`. `nonebasisvc` calls `makebasis` with its argument.
The basis is appropriate for Hamiltonians without any conserved quantities, and for checking symmetry-adapted calculations.
`nonebasis` gives the same basis in the creation operator representation.

Example.

```
snegfermionoperators[d];
nonebasisvc[{d[]}] // InputForm
```

normop

`normop[a]` gives the norm $\sqrt{\langle a|a\rangle}$ of the state $|a\rangle = a|0\rangle$ in the creation-operator representation.
`normop[a]` is the square root of `scalarproductop[a, a]`. The vacuum is the one set by `ordering`.
`normop` is `Listable`.

Example. Normalization of the singlet state of two electrons:

```
snegfermionoperators[c];
s = nc[c[CR, 1, UP], c[CR, 2, DO]] - nc[c[CR, 1, DO], c[CR, 2, UP]];
normop[s]
```

normvc

`normvc[v]` gives the norm $\sqrt{\langle v|v\rangle}$ of the state v in the occupation number representation.
`normvc[v]` is `Sqrt[vev[scalarproductvc[v, v]]]`.
`normvc` is `Listable`. The corresponding function for states in the creation operator representation is `normop`.
Normalized states are obtained as `v/normvc[v]`.

Example.

```
snegfermionoperators[d];
makebasis[{d[]}];
normvc[3 vc[1, 0] + 4 I vc[0, 1]]
```

op2ndx

`op2ndx[c[t, i, σ]]` gives the position of the operator $c_{i\sigma}$ in the single-particle basis `BASIS` defined by `makebasis`.
The position is the index of the occupation number of the state $c^\dagger_{i\sigma}|0\rangle$ in the `vc` vectors.
The creation/annihilation index `t` (`CR` or `AN`) is disregarded: `op2ndx[c[AN, i, σ]]` equals `op2ndx[c[CR, i, σ]]`.
If the operator is not in `BASIS`, `op2ndx` returns unevaluated; see `goodopQ`.
`op2ndx` is `Listable`. The inverse function is `ndx2op`.

Example.

```
snegfermionoperators[c, d];
makebasis[{c[], d[]}];
{op2ndx[c[CR, UP]], op2ndx[c[AN, DO]], op2ndx[d[CR, UP]], op2ndx[d[AN,
    DO]]}
```

ops2vc

`ops2vc[ops]` converts the state $\hat{O}|0\rangle$, given by the operator expression *ops* acting on the vacuum, into the occupation number representation.
`ops2vc[ops, v]` gives $\hat{O}|v\rangle$ for a state v in the occupation number representation.
`ops2vc[ops]` is `ap[ops, vacuum[]]`, and `ops2vc[ops, v]` is `ap[ops, v]`.
The operators must belong to the single-particle basis `BASIS` defined by `makebasis`.

`ops2vc` is the inverse of `vc2ops`. `bzop2bzvc` converts a whole basis.

Example.

```
snegfermionoperators[c];
makebasis[{c[1], c[2]}];
ops2vc[nc[c[CR, 2, DO], c[CR, 1, UP]]] // InputForm
```

orthogbasisop

`orthogbasisop[bz1, bz]` orthonormalizes the states of every subspace of the basis *bz1* with respect to the states of the corresponding subspace of the basis *bz*; both bases are in the creation operator representation.
`orthogbasisop[bz1, bz]` is `mapbasis[bz1, bz, orthogop]`; see `orthogbasisvc` for the conventions. The subspaces of *bz1* and *bz* must have the same quantum numbers in the same order.
`transformtoLR` uses `orthogbasisop` to orthonormalize the symmetric and antisymmetric combinations of the basis states under the mirror symmetry.

Example.

```
snegfermionoperators[d];
bz = qszbasis[{d[]}];
orthogbasisop[applybasis[bz, 3 # &], bz]
```

orthogbasisvc

`orthogbasisvc[bz1, bz]` orthonormalizes the states of every subspace of the basis *bz1* with respect to the states of the corresponding subspace of the basis *bz*; both bases are in the occupation number representation.
`orthogbasisvc[bz1, bz]` is `mapbasis[bz1, bz, orthogvc]`: for each subspace, `orthogvc[states1, states]` is computed. The subspaces of *bz1* and *bz* must have the same quantum numbers in the same order; otherwise the message `mapbasis::compat` is issued and `Null` is returned.
Typically, *bz1* is obtained from the orthonormal basis *bz* by applying an operator (projector, symmetrizer) to the states. The result is an orthonormal basis of the transformed states in each subspace; components outside the span of the original subspace are discarded. Subspaces can become empty; remove them with `dropemptysubspaces`.
`orthogbasisop` is the corresponding function for the creation operator representation.

Example.

```
snegfermionoperators[d];
bz = qszbasisvc[{d[]}];
```

```
bz1 = applybasis[bz, 2 # &];
orthogbasisvc[bz1, bz] // InputForm
```

orthogop

`orthogop[vecs, basis]` orthonormalizes the states *vecs* in the creation-operator representation, using the orthonormal basis states *basis*.
Every state in *vecs* is decomposed in *basis* with `decomposeop` (scalar products `scalarproductop`), the coefficient vectors are orthonormalized in the order of *vecs* (Gram-Schmidt), and the resulting states are expanded in *basis*.
Linearly dependent states are dropped, so the result spans the same subspace as *vecs* and may be shorter.
basis must be a list of orthonormal states that spans *vecs*. The vacuum is the one set by `ordering`.

Example. Orthonormalization of the two-electron states with one electron on each site:

```
snegfermionoperators[c];
b = {nc[c[CR, 1, UP], c[CR, 2, DO]], nc[c[CR, 1, DO], c[CR, 2, UP]]};
orthogop[{b[[1]] - b[[2]], b[[1]] + b[[2]], b[[1]]}, b]
```

orthogvc

`orthogvc[vecs, basis]` gives an orthonormal basis of the space spanned by the states *vecs*, expressed in the orthonormal states *basis*; all states are in the occupation number representation.
Each state of *vecs* is decomposed into its components $\langle b_j|v\rangle$ with respect to the states b_j of *basis* (`decomposevc`). The coefficient vectors are orthonormalized with the Gram-Schmidt procedure (`Orthogonalize` with the inner product $\sum_j a_j^* b_j$ and `Simplify`), vectors that are proven to be zero are dropped, and the results are converted back to states.
The result is a list of orthonormal states; its length is the dimension of the span of *vecs*. Linearly dependent states are removed.
basis must be orthonormal. Components of *vecs* outside the span of *basis* are discarded: `orthogvc` effectively projects onto the span of *basis* before orthogonalizing.
Symbolic coefficients are supported; the coefficients are simplified with `Simplify`, which can be slow for complicated expressions.
`orthogop` is the corresponding function for the creation operator representation.
`transformbasis` uses `orthogvc` after applying an operator (e.g. a projector) to the states of each subspace.

Example.

```
snegfermionoperators[d];
makebasis[{d[]}];
orthogvc[{vc[1, 0] + vc[0, 1], vc[1, 0]}, {vc[1, 0], vc[0, 1]}] //
    InputForm
```

phononbasis

`phononbasis[nmax]` gives the oscillator states `ket[0]` through `ket[nmax]`.
`phononbasis[{n1, n2, ...}]` gives the tensor-product kets of several oscillators.
Cutoffs are nonnegative integers and include the upper state. The total dimension is the product of the individual cutoff-plus-one factors. Occupations are ordered with the last mode varying fastest.
Use `transformtoPH` to attach these kets to each state of an electronic basis.

Example.

```
phononbasis[{1,2}]
```

phononid

`phononid[nmax]` gives the identity on the oscillator subspace with occupations zero through *nmax*.
Written in ket-bra form, this is $\sum_{n=0}^{n_{\max}} |n\rangle\langle n|$. It acts as a projector if applied to states outside that subspace.
Use an explicit identity on a spectator oscillator when constructing a tensor product of local operators.

Example.

```
nc[phononid[2], #] & /@ {ket[0],ket[2],ket[3]}
```

phononminus

`phononminus[nmax]` lowers the oscillator occupation by one, with matrix element $\sqrt{n}$.
`phononminus[i, cutoffs]` lowers mode *i* in the product space with the given inclusive cutoffs.
The vacuum is annihilated. The finite-dimensional commutator with `phononplus` has diagonal entries one except on the top state, where it equals minus the cutoff. The trace of the commutator is therefore zero, as for any finite matrices.

Example.

```
nc[phononminus[2],#] & /@ phononbasis[2]
```

phononnumber

`phononnumber[nmax]` gives the number operator of one truncated oscillator.
`phononnumber[i, cutoffs]` counts quanta in mode i (numbered from one).
Its eigenvalue on a retained occupation ket is the corresponding occupation. The free oscillator energy is the frequency times this operator when the zero-point energy is dropped and $\hbar = 1$.
The product of the projected creator and annihilator is exactly the projected number operator, including on the top state. The opposite product is not the number plus identity there.

Example.

```
nc[phononnumber[2,{1,3}],ket[1,2]]
```

phononplus

`phononplus[nmax]` creates an oscillator quantum in a space cut off inclusively at *nmax*.
`phononplus[i, cutoffs]` raises mode i of a product of oscillators.
On $|n\rangle$ the matrix element is $\sqrt{n+1}$ if the new state is retained, and zero at the upper cutoff. This is the projected creator, not the unbounded boson of `snegbosonoperators`.
Modes are numbered from one; the operator acts as the identity on the other truncated mode spaces.

Example.

```
nc[phononplus[2],#] & /@ phononbasis[2]
```

phononx

`phononx[nmax]` gives the truncated dimensionless oscillator displacement $b + b^\dagger$.
`phononx[i, cutoffs]` gives the displacement of mode i in a tensor-product basis.
For oscillator mass M and angular frequency ω, multiply by $\sqrt{\hbar/(2M\omega)}$ to obtain a physical displacement. There is no factor of $1/\sqrt{2}$ in the SNEG operator.
A linear coupling mixes adjacent occupation states. Check convergence with cutoff in displaced-oscillator calculations; the top state has no raising matrix element.

Example.

In[1]:=
```
Table[nc[bra[i],phononx[2],ket[j]], {i,0,2}, {j,0,2}] // MatrixForm
```

Out[1]=
$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & \sqrt{2} \\ 0 & \sqrt{2} & 0 \end{pmatrix}$$

qbasis

`qbasis[{c[1], c[2], ...}]` gives the basis of the Fock space of the levels `c[1]`, `c[2]`, ... in the creation operator representation, organized in subspaces with well-defined charge Q.
`qbasis[ops]` is `bzvc2bzop[qbasisvc[ops]]`; see `qbasisvc` for the definition $Q = N - N_s/2$. `qbasis` calls `makebasis` with its argument.
NRG Ljubljana uses `qbasis` as the starting point for the symmetry types `U1`, `SL`, and `SU2` (isospin, after `transformQStoIS`), and for the charge-parity types `P` and `PP`, where Q is replaced by ± 1 according to the parity of Q and the subspaces are merged with `mergebasis`.

Example.

```
snegfermionoperators[d];
qbasis[{d[]}]
```

qbasisvc

`qbasisvc[{c[1], c[2], ...}]` gives the basis of the Fock space of the levels `c[1]`, `c[2]`, ... in the occupation number representation, organized in subspaces with well-defined charge Q.
The result is a list `{{{Q}, {states}}, ...}`; the states are `vc` vectors in lexicographic order. `qbasisvc` calls `makebasis` with its argument.
$Q = N - N_s/2$, where N_s is the number of single-particle states (the length of `BASIS`); for spin-1/2 fermions, $Q = N - N_{\text{levels}}$. The subspace Q has $\binom{N_s}{N}$ states.
`qbasisvc` works for fermions with any spin, including spinless fermions. It is the appropriate basis for Hamiltonians that conserve only the particle number, e.g. with spin-orbit coupling or with a magnetic field in a general direction.
`qbasis` gives the same basis in the creation operator representation.

Example.

```
snegfermionoperators[d];
qbasisvc[{d[]}]
```

qsbasis

`qsbasis[{c[1], c[2], ...}]` gives a basis of the Fock space of the spin-1/2 levels `c[1]`, `c[2]`, ... in the creation operator representation, organized in subspaces with

well-defined charge Q and total spin S; each subspace contains the states with $S_z = S$.
The states are strings of creation operators acting on the vacuum (1 is the vacuum). The construction, the quantum numbers $Q = N - N_{\text{levels}}$ and S, and the phase conventions are the same as for `qsbasisvc`; `bzop2bzvc[qsbasis[ops]]` equals `qsbasisvc[ops]`.
`qsbasis` calls `makebasis` with its argument. The operators must be spin-1/2 fermion operators; otherwise the message `qsbasis::spin` is issued and `$Failed` is returned.
`qsbasis[{}]` is `{{{0, 0}, {1}}}`.
The construction in the creation operator representation uses `spindown`, which normalizes states with `vev`; it is slower than `qsbasisvc` for more than a few levels.
`transformtoLR` (mirror parity) and `SU2basistensorproduct` expect bases in this representation.

Example.

```
snegfermionoperators[d];
qsbasis[{d[]}]
```

qsbasisvc

`qsbasisvc[{c[1], c[2], ...}]` gives a basis of the Fock space of the spin-1/2 levels `c[1]`, `c[2]`, ... in the occupation number representation, organized in subspaces with well-defined charge Q and total spin S; each subspace contains the states with $S_z = S$.
The result is a list `{{{Q, S}, {states}}, ...}` of invariant subspaces of the group $U(1)_{\text{charge}} \times SU(2)_{\text{spin}}$; the states are linear combinations of `vc` vectors. `qsbasisvc` calls `makebasis` with its argument.
The charge is measured from half filling, $Q = N - N_{\text{levels}}$.
Only one state of each spin multiplet is kept: the highest-weight state with $S_z = S$. Every state in the subspace (Q, S) represents a multiplet of $2S + 1$ degenerate states, which are obtained by applying the total spin-lowering operator S^-. Hence $\sum_{(Q,S)} (2S + 1)\, d_{Q,S} = 4^{N_{\text{levels}}}$, where $d_{Q,S}$ is the number of states in the subspace.
Matrices computed in this basis (`makematricesbzvc`) are the reduced matrices of operators that commute with all components of the total spin, $[H, \mathbf{S}] = 0$: each eigenvalue in the subspace (Q, S) is $(2S + 1)$-fold degenerate. For operators that do not commute with $S^{\pm}$, such as a Zeeman term, use `qszbasisvc`.
Construction: the levels are added one at a time, in the order of the list. If $|a\rangle$ is a state of the first levels with quantum numbers (q, s) and $S_z = s$, adding the level c gives the states

$$(q-1, s) : |a\rangle, \quad (q+1, s) : c_\downarrow^\dagger c_\uparrow^\dagger |a\rangle, \quad (q, s+\tfrac{1}{2}) : c_\uparrow^\dagger |a\rangle,$$

$$(q, s-\tfrac{1}{2}):\ \sqrt{\frac{2s}{2s+1}}\,c_\downarrow^\dagger|a\rangle - \sqrt{\frac{1}{2s+1}}\,c_\uparrow^\dagger|a'\rangle \quad (s \geq \tfrac{1}{2}),$$

where $|a'\rangle = S^-|a\rangle/\|S^-|a\rangle\|$ has $S_z = s-1$. The coefficients are the Clebsch-Gordan coefficients for coupling the spin s of the existing states with the spin 1/2 of the added electron, with the Condon-Shortley phase convention; the operator of the added level stands to the left of the operators of the previous levels. This fixes the phases of the basis states; for example, the one-level states are $|0\rangle$, $c_\uparrow^\dagger|0\rangle$, and $c_\downarrow^\dagger c_\uparrow^\dagger|0\rangle = -$`vc[1, 1]`.
The operators must be spin-1/2 fermion operators (declared with `snegfermionoperators`). Otherwise the message `qsbasisvc::spin` is issued and `$Failed` is returned. For other fermions, use `qszbasisvc`, `qbasisvc`, or `nonebasisvc`. `qsbasisvc[{}]` is `{{{0, 0}, {vc[]}}}`.
The subspaces are sorted by (Q, S). The states are simplified with `Simplify`. `qsbasis` gives the same basis in the creation operator representation; `sbasisvc` merges the subspaces with different Q. Bases with isospin are obtained with `quickISObasis` and `transformQStoIS`.

Example. The basis of a single level. The subspace $(0, 1/2)$ contains only the state with $S_z = +1/2$:

```
snegfermionoperators[d];
qsbasisvc[{d[]}] // InputForm
```

QSZaddspinket

`QSZaddspinket[bvc, s]` adds a spin s to the basis *bvc* with the quantum numbers (Q, S_z), giving the tensor-product basis with the states `vc[..., ket[m]]`. `QSZaddspinket[bvc, {s1, s2, ...}]` adds several spins, with the states `vc[..., ket[m1, m2, ...]]`.
bvc is a basis in the occupation-number representation with subspaces labeled `{Q, Sz}`, as given by `qszbasisvc`. The result has the subspaces `{Q, Sz + m}` with m the total projection of the added spins, so that S_z is the projection of the total spin (fermions and impurity spins). The order of the subspaces follows from `mergebasis`. The spins occupy the slots of the kets in the given order; operators on them are obtained with `spinketbrarules`, applied to an expression in abstract spin operators, or with `spinketbraX`, ... with position arrays.
The key of each subspace must be of the form `{Q, Sz}`; the last element is taken as S_z. For bases with other quantum numbers the result is not meaningful.
NRG Ljubljana uses this construction for the initial Hamiltonian of quantum impurity models with impurity spins, e.g. the Kondo model.

Example. A spin 1/2 added to the basis of one fermion level:

```
snegfermionoperators[c];
QSZaddspinket[qszbasisvc[{c[]}], 1/2]
```

qszbasis

`qszbasis[{c[1], c[2], ...}]` gives the basis of the Fock space of the levels `c[1]`, `c[2]`, ... in the creation operator representation, organized in subspaces with well-defined charge Q and spin projection S_z.
`qszbasis[{c[1], ...}, 1/2]` uses the specialized implementation for spin-1/2 fermions.
`qszbasis[ops]` is `bzvc2bzop[qszbasisvc[ops]]`: the states are strings of creation operators (in the normal order of `nc`) that act on the vacuum; 1 is the vacuum. See `qszbasisvc` for the definition of the quantum numbers, $Q = N - N_s/2$ and S_z, and for the order of the subspaces.
`qszbasis` calls `makebasis` with its argument.
Matrices of operators in this basis are obtained with `makematricesbzop`; `bzop2bzvc` converts the basis to the occupation number representation.

Example.

```
snegfermionoperators[d];
qszbasis[{d[]}]
```

qszbasisvc

`qszbasisvc[{c[1], c[2], ...}]` gives the basis of the Fock space of the levels `c[1]`, `c[2]`, ... in the occupation number representation, organized in subspaces with well-defined charge Q and spin projection S_z.
`qszbasisvc[{c[1], ...}, 1/2]` uses a specialized implementation for spin-1/2 fermions.
The result is a list `{{{Q, Sz}, {states}}, ...}` of invariant subspaces; the states are `vc` vectors. This is the standard form of a basis in SNEG (see `makematricesbzvc`, `bzQ`).
`qszbasisvc` calls `makebasis` with its argument; the `vc` vectors refer to this `BASIS`. The levels are given without the creation/annihilation and spin indexes.
The charge is measured from half filling: $Q = N - N_s/2$, where N is the number of particles and N_s the number of single-particle states (the length of `BASIS`). For spin-1/2 fermions, $N_s/2$ is the number of levels, so that $Q = N - N_{\text{levels}}$; the empty system has $Q = -N_{\text{levels}}$, and the half-filled system $Q = 0$.
The spin projection is $S_z = \sum_k m_k n_k$, where $m_k = \pm 1/2$ for the states `UP` and `DO` of spin-1/2 fermions; for fermions with spin s, m_k is the spin index of the k-th element of `BASIS`.

The basis states are the occupation number states `vc[n1, n2, ...]`, which are eigenstates of Q and S_z; see `vc` for the sign convention. The subspaces are sorted by (Q, S_z); within a subspace, the states are in lexicographic order of the occupation numbers. The specialized implementation `qszbasisvc[ops, 1/2]` gives the same subspaces with a different order of the states.
The total number of states is 2^{N_s}, i.e. $4^{N_{\text{levels}}}$ for spin-1/2 fermions.
The (Q, S_z) basis is appropriate for Hamiltonians that conserve the particle number and S_z, e.g. in a magnetic field along z or with spin anisotropy along z. For SU(2) spin-symmetric Hamiltonians, `qsbasisvc` gives smaller subspaces.
`qszbasis` gives the same basis in the creation operator representation.

Example. The basis of a single spin-1/2 level:

```
snegfermionoperators[d];
qszbasisvc[{d[]}]
```

quickDBL

`quickDBL[ops1, ops2, fnc]` gives the product basis of two subsystems with the levels *ops1* and *ops2*, with the quantum numbers `{X1, X2}`, where `X1` and `X2` are the first quantum numbers of the bases `fnc[ops1]` and `fnc[ops2]`.
fnc is a basis generator in the creation operator representation, such as `qbasis`, `qszbasis`, or `quickSU2basis`. Only the first quantum number of each subsystem basis is kept; the subspaces with equal first quantum numbers are merged before the product is formed.
The product states are `nc[state1, state2]` (the operator strings of the first subsystem to the left), and the quantum numbers are joined: `{X1, X2}`. The subspaces are not merged further.
`quickDBL` calls `makebasis[Join[ops1, ops2]]` at the end, so that `bzop2bzvc` can be applied to the result.
The basis is appropriate for systems with two separately conserved quantities, e.g. the charges Q_1, Q_2 of two decoupled channels (`qbasis`), or the isospins I_1, I_2 of two particle-hole symmetric channels (`quickSU2basis`, the `DBLSU2` symmetry type of NRG Ljubljana).
For SU(2) quantum numbers (isospin), the product of highest-weight states is the highest-weight state of the product of the two multiplets: each state represents $(2I_1 + 1)(2I_2 + 1)$ states. Generators whose subspaces contain only highest-weight states of a quantum number that is dropped (e.g. `qsbasis`, where S is dropped) give incomplete bases; see Possible Issues.

Example. Two levels with separately conserved charges:

```
snegfermionoperators[c];
quickDBL[{c[1]}, {c[2]}, qbasis]
```

quickDBLSZ

`quickDBLSZ[ops1, ops2, fnc]` gives the product basis of two subsystems with the levels *ops1* and *ops2*, with the quantum numbers `{X1, X2, Sz}`, where `{X1, Sz1}` and `{X2, Sz2}` are the quantum numbers of the bases `fnc[ops1]` and `fnc[ops2]`, and $S_z = S_{z1} + S_{z2}$.
fnc is a basis generator in the creation operator representation with two quantum numbers, the second being S_z: `qszbasis` (quantum numbers Q_1, Q_2, S_z, the `DBLQSZ` symmetry type of NRG Ljubljana) or `quickISOSZbasis` (I_1, I_2, S_z, `DBLISOSZ`).
The product states are `nc[state1, state2]`; the subspaces with equal `{X1, X2, Sz}` are merged with `mergebasis`.
`quickDBLSZ` calls `makebasis[Join[ops1, ops2]]` at the end, so that `bzop2bzvc` can be applied to the result.

Example.

```
snegfermionoperators[c];
bz = quickDBLSZ[{c[1]}, {c[2]}, qszbasis];
{First[#], Length[Last[#]]} & /@ bz
```

quickISObasis

`quickISObasis[{c[1], c[2], ...}]` gives a basis of the Fock space of the spin-1/2 levels `c[1]`, `c[2]`, ... in the creation operator representation, organized in subspaces with well-defined total isospin I and total spin S; each subspace contains the states with $I_z = I$ and $S_z = S$.
The total isospin is $\mathbf{I} = \sum_i \mathbf{I}_i(n_i)$, with $I^z = Q/2 = (N - N_{\text{levels}})/2$ and $I^+ = \sum_i (-1)^{n_i} c^\dagger_{i\uparrow} c^\dagger_{i\downarrow}$ (see `isospin`). The phase exponent n_i of the level `c[i]` is the value of `nnop[c[i]]`; for a bipartite lattice, it is even on one sublattice and odd on the other.
`nnop[c[i]]` must be defined for all levels before `quickISObasis` is called (see Possible Issues).
The result is a list `{{{I, S}, {states}}, ...}`. Each state represents a multiplet of $(2I+1)(2S+1)$ states, obtained by applying I^- and S^-; hence $\sum (2I+1)(2S+1)\, d_{I,S} = 4^{N_{\text{levels}}}$.
Matrices computed in this basis are the reduced matrices of operators that commute with both $\mathbf{S}$ and $\mathbf{I}$, such as the half-filled Hubbard model in the particle-hole symmetric form $H = -t \sum_{\langle ij\rangle\sigma} (c^\dagger_{i\sigma} c_{j\sigma} + \text{h.c.}) + \frac{U}{2} \sum_i (n_i - 1)^2$ on a bipartite lattice, or the particle-hole symmetric Anderson and Kondo impurity models. This $SU(2)_{\text{spin}} \times SU(2)_{\text{isospin}}$ symmetry (the `ISO` symmetry type of NRG Ljubljana) reduces the matrix sizes considerably.

Construction: the (Q, S) basis of `qsbasisvc` is transformed with `transformQStoIS`: starting from the highest charge, the states of each subspace are the isospin highest-weight states with $I = Q/2$; the multiplet partners generated by I^- are projected out of the subspaces with lower Q. The result is converted to the creation operator representation.
`quickISObasis` calls `makebasis` with its argument. Use `bzop2bzvc` to obtain the occupation number representation.
The subspaces are ordered by decreasing Q of the highest-weight states, not sorted. `quickISOSZbasis` gives the (I, S_z) basis, and `quickSU2basis` the basis with isospin only.

Example. A single level: the empty and doubly occupied states form an isospin doublet, represented by the doubly occupied state, and the singly occupied states form a spin doublet:

```
snegfermionoperators[d];
nnop[d[]] = 0;
quickISObasis[{d[]}]
```

quickISOSZbasis

`quickISOSZbasis[{c[1], c[2], ...}]` gives a basis of the Fock space of the levels `c[1]`, `c[2]`, ... in the creation operator representation, organized in subspaces with well-defined total isospin I and spin projection S_z; each subspace contains the states with $I_z = I$.
`quickISOSZbasis` applies `transformQStoIS` to the (Q, S_z) basis of `qszbasisvc`. The result is `{{{I, Sz}, {states}}, ...}`; each state represents an isospin multiplet of $2I + 1$ states.
The isospin phases are taken from `nnop`, which must be defined for all levels; see `quickISObasis` for the definitions, the conventions, and the pitfalls.
The basis is appropriate for particle-hole symmetric Hamiltonians that conserve S_z but not the total spin, e.g. with a magnetic field along z (the `ISOSZ` symmetry type of NRG Ljubljana).

Example.

```
snegfermionoperators[d];
nnop[d[]] = 0;
quickISOSZbasis[{d[]}]
```

quickSU2basis

`quickSU2basis[{c[1], c[2], ...}]` gives a basis of the Fock space of the levels `c[1]`, `c[2]`, ... in the creation operator representation, organized in subspaces with

well-defined total isospin I; each subspace contains the states with $I_z = I$.
`quickSU2basis` applies `transformQStoIS` to the charge basis of `qbasisvc`. The result is `{{{I}, {states}}, ...}`; each state represents an isospin multiplet of $2I + 1$ states.
The isospin phases are taken from `nnop`, which must be defined for all levels; see `quickISObasis` for the definitions, the conventions, and the pitfalls.
The basis is appropriate for particle-hole symmetric Hamiltonians without spin symmetry, e.g. with spin-orbit coupling or a magnetic field in an arbitrary direction (the `SU2` symmetry type of NRG Ljubljana).

Example.
```
snegfermionoperators[d];
nnop[d[]] = 0;
quickSU2basis[{d[]}]
```

remove1state

`remove1state[vecs, s]` gives an orthonormal basis of the orthogonal complement of the state s in the space spanned by the orthonormal states *vecs*; all states are in the occupation number representation.
Each state v of *vecs* is replaced by $v - \langle s|v\rangle\, s$, and the results are orthonormalized with `orthogvc[..., vecs]`. The result has one state fewer than *vecs* if s lies in the span of *vecs*.
vecs and s must be normalized; *vecs* must be orthonormal.
`remove1state[{}, s]` is `{}`. `removestates` removes several states.
`transformQStoIS` uses `removestates` to project the isospin multiplet partners out of the subspaces with lower charge.

Example. Removing the triplet state $(|\uparrow\downarrow\rangle + |\downarrow\uparrow\rangle)/\sqrt{2}$ from the two-site states with $S_z = 0$ and single occupancy leaves the singlet:
```
snegfermionoperators[c];
makebasis[{c[1], c[2]}];
remove1state[{vc[1, 0, 0, 1], vc[0, 1, 1, 0]}, (vc[1, 0, 0, 1] + vc[0,
    1, 1, 0])/Sqrt[2]] // InputForm
```

removestates

`removestates[vecs, states]` gives an orthonormal basis of the orthogonal complement of the normalized states *states* in the space spanned by the orthonormal states *vecs*; all states are in the occupation number representation.

`removestates` applies `remove1state` successively for each state of *states*: `Fold[remove1state, vecs, states]`. For the result to be the orthogonal complement, the states in *states* must be mutually orthogonal.

Example.

```
snegfermionoperators[d];
makebasis[{d[]}];
removestates[{vc[0, 0], vc[1, 0], vc[0, 1]}, {(vc[1, 0] + vc[0, 1])/Sqrt
    [2], vc[0, 0]}] // InputForm
```

sbasis

`sbasis[{c[1], c[2], ...}]` gives a basis of the Fock space of the spin-1/2 levels `c[1]`, `c[2]`, ... in the creation operator representation, organized in subspaces with well-defined total spin S; each subspace contains the states with $S_z = S$.
`sbasis[ops]` is `bzvc2bzop[sbasisvc[ops]]`; see `sbasisvc` and `qsbasisvc`. `sbasis[{}]` is `{{{0}, {1}}}`.
NRG Ljubljana uses `sbasis` for the symmetry types `SPSU2` and `SPSU2LR` (spin symmetry without charge conservation).

Example.

```
snegfermionoperators[d];
sbasis[{d[]}]
```

sbasisvc

`sbasisvc[{c[1], c[2], ...}]` gives a basis of the Fock space of the spin-1/2 levels `c[1]`, `c[2]`, ... in the occupation number representation, organized in subspaces with well-defined total spin S; each subspace contains the states with $S_z = S$.
`sbasisvc` constructs the (Q, S) basis with `qsbasisvc`, drops Q, and merges the subspaces with equal S. The result is `{{{S}, {states}}, ...}`; each state represents a multiplet of $2S + 1$ states.
The basis is appropriate for spin-symmetric Hamiltonians that do not conserve the particle number, e.g. with superconducting pairing. The operators must be spin-1/2 fermions; see `qsbasisvc` for the conventions.
`sbasisvc[{}]` is `{{{0}, {vc[]}}}`. `sbasis` gives the same basis in the creation operator representation.

Example.

```
snegfermionoperators[d];
sbasisvc[{d[]}] // InputForm
```

scalarproduct

`scalarproduct[a, b]` gives the scalar product $\langle a|b\rangle$ of states in the occupation number representation; it is an alias of `scalarproductvc`.
`scalarproduct[a, b]` evaluates to `scalarproductvc[a, b]`; see that page for the conventions. The first argument is conjugated.
For states in the creation operator representation, use `scalarproductop`.

Example.

```
snegfermionoperators[d];
makebasis[{d[]}];
scalarproduct[(vc[1, 0] + vc[0, 1])/Sqrt[2], vc[1, 0]]
```

scalarproductop

`scalarproductop[a, b]` gives the scalar product $\langle a|b\rangle$ of the states $|a\rangle = a|0\rangle$ and $|b\rangle = b|0\rangle$ given in the creation-operator representation.
In the creation-operator representation, a state is an operator expression acting on the vacuum, e.g. `nc[c[CR, 1, UP], c[CR, 2, DO]]` for $c^\dagger_{1\uparrow}c^\dagger_{2\downarrow}|0\rangle$.
The result is $\langle 0|a^\dagger b|0\rangle$, computed as `vevwick[nc[conj[a], zeroonvac[b]]]`. The first argument is conjugated: `scalarproductop` is antilinear in a and linear in b.
The vacuum is the one set by `ordering`: the empty state or, for `ordering[c] = SEA`, the Fermi sea.
Terms of b that annihilate the vacuum are dropped by `zeroonvac`.
`scalarproductop` is `Listable`.

Example. Overlaps of two-electron states on two sites. The sign depends on the order of the creation operators:

```
snegfermionoperators[c];
s = nc[c[CR, 1, UP], c[CR, 2, DO]] - nc[c[CR, 1, DO], c[CR, 2, UP]];
t0 = nc[c[CR, 1, UP], c[CR, 2, DO]] + nc[c[CR, 1, DO], c[CR, 2, UP]];
{scalarproductop[s, s], scalarproductop[s, t0], scalarproductop[t0, nc[c
    [CR, 2, DO], c[CR, 1, UP]]]}
```

scalarproductvc

`scalarproductvc[a, b]` gives the scalar product $\langle a|b\rangle$ of the states a and b in the occupation number representation.
a and b are `vc` vectors or linear combinations of them. The `vc` vectors are orthonormal: `scalarproductvc[vc[...], vc[...]]` is 1 if both vectors are identical and 0 otherwise.

Physics convention: the scalar product is conjugate-linear in the first argument and linear in the second, $\langle \alpha a | \beta b \rangle = \alpha^* \beta \langle a | b \rangle$. Complex constants declared with `snegcomplexconstants` are conjugated; real constants and numbers are handled accordingly.
For states with kets, `vc[..., ket[k1]]` and `vc[..., ket[k2]]`, the fermionic overlap is multiplied by `braketrule[bra[k1], ket[k2]]`.
In an `nc` product, `nc[conj[v1], v2]` with `vc` vectors *v1* and *v2* is replaced by `scalarproductvc[v1, v2]`.
`scalarproductvc` is `Listable`. `scalarproduct` is an alias.
The vectors are compared literally; symbolic occupation numbers are not supported.
Undeclared symbols in the coefficients are treated as operators, and the scalar product then remains partly unevaluated (see Possible Issues).

Example.

```
snegfermionoperators[d];
makebasis[{d[]}];
{scalarproductvc[vc[1, 0], vc[1, 0]], scalarproductvc[vc[1, 0], vc[0,
    1]]}
```

showbzmat

`showbzmat[mats]` displays a list of matrices `{{qn, mat}, ...}` in invariant subspaces as a table of the quantum numbers and the matrices in `MatrixForm`.
mats is typically the result of `makematricesbzvc` or `makematricesbzop`. The argument must satisfy `bzQ`; otherwise `showbzmat` stays unevaluated.
The result is wrapped in `MatrixForm` and is meant for display only.

Example. Hamiltonian of the Hubbard atom with a magnetic field in the (Q, S_z) subspaces:

```
snegfermionoperators[d];
snegrealconstants[ε, U, B];
showbzmat[makematricesbzvc[ε number[d[]] + U hubbard[d[]] + B spinz[d
    []], qszbasisvc[{d[]}]]]
```

spinbasis

`spinbasis[s]` gives the basis of a spin *s* in the bra-ket representation, as a list of subspaces `{{m}, {ket[m]}}` labeled by the spin projection.
`spinbasis[{s1, s2, ...}]` gives the product basis of several spins, `ket[m1, m2, ...]`, organized in subspaces of total $S_z = \sum_k m_k$.
`spinbasis[s, pos]` places a single spin in the slots specified by the position array *pos*.

The result has the form of the bases used by `makematricesbzop`: a list of `{{qn}, {states}}`.
For a single spin, the subspaces are ordered from $m = s$ down to $m = -s$. For several spins, the subspaces are ordered by increasing total S_z; spin k occupies the k-th slot of the kets.
`QSZaddspinket` combines a spin basis with a fermionic basis with good charge and spin projection.
All spins must be integers or half-integers.

Example.
```
spinbasis[1]
```

spinbra

`spinbra[s]` gives the list of the spin-s basis bras `bra[m]`, $m = s, s-1, \ldots, -s$.
`spinbra[s, pos]` places the spin projection in the slots specified by the position array *pos*.
`spinbra[s, pos]` is `conj[spinket[s, pos]]`; see `spinket`.

Example.
```
spinbra[1/2]
```

spindownvc

`spindownvc[v]` applies the total spin-lowering operator $S^- = \sum_j c^\dagger_{j\downarrow} c_{j\uparrow}$ to the state v in the occupation-number representation and normalizes the result.
v is a vector `vc[n1, n2, ...]` or a linear combination of such vectors, in the basis declared with `makebasis`. The occupation numbers are taken in consecutive pairs (spin up, spin down) of the levels, the order of `makebasis` for spin-1/2 fermions. Each pair `1, 0` (spin up) is replaced by `0, 1` (spin down). Since the up and down operators of a level are adjacent in the basis, no fermion sign arises. The result is divided by its norm; if $S^- v = 0$, it is 0.
A spin ket at the end of the vector, `vc[..., ket[m]]` (see `QSZaddspinket`), is not lowered: only the fermionic part is acted upon, and the result is 0 if that vanishes.

Example.
```
snegfermionoperators[c];
makebasis[{c[1], c[2]}];
v = ops2vc[nc[c[CR, 1, UP], c[CR, 2, UP]]]
```

spinket

`spinket[s]` gives the list of the spin-s basis kets `ket[m]`, $m = s, s-1, \ldots, -s$.
`spinket[s, pos]` places the spin projection in the slots of the kets specified by the position array *pos*; the other slots are `Null`.
The kets are orthonormal, $\langle m|m'\rangle = \delta_{mm'}$ (see `braketrule`). `spinbra` gives the conjugate bras, and `spinbasis` the kets in the form of a basis with the quantum number S_z.
The position array is a list of 0 and 1; e.g. `{1, 0}` gives `ket[m, Null]` for the first of two spins. See `spinketbraX`.
s must be an integer or a half-integer.

Example.
```
spinket[1]
```

spinketbraI

`spinketbraI[s]` gives the identity operator $\sum_m |m\rangle\langle m|$ of the spin-s space in the bra-ket representation.
`spinketbraI[s, pos]` places it in the slots specified by the position array *pos*.
The identity is needed to extend an operator that acts on one slot of a multi-spin ket to the full ket, e.g. $S^z_1 \otimes 1_2$ = `nc[spinketbraZ[s1, {1, 0}], spinketbraI[s2, {0, 1}]]`. `spinketbrarules` includes these factors automatically.
See `spinketbraX` for the position arrays.

Example.
```
spinketbraI[1]
```

spinketbrarules

`spinketbrarules[{S[1], S[2], ...}]` gives replacement rules that map the abstract spin operators of the listed sites to their bra-ket representation.
`spinketbrarules[{S -> {j1, j2, ...}}]` specifies the sites by their indexes.
For each listed site and each component $t = 1, \ldots, 5$ (S^x, S^y, S^z, S^+, S^-) a rule `S[j..., t] -> op` is generated. The k-th site in the list corresponds to the k-th slot of the kets `ket[m1, m2, ...]`, as in `spinbasis[{s1, s2, ...}]` and `QSZaddspinket[bvc, {s1, s2, ...}]`.
Each operator includes the identities `spinketbraI` of the other slots, so that it acts on the full kets. The number of terms therefore grows as $\prod_k (2s_k + 1)$.
All spins must have a declared numeric size. A site without indexes is given as `S[]`, as for a single impurity spin with the operators `S[1]`, ..., `S[5]`. Site 0 must be given

as `S -> {0}`, since `S[0]` evaluates to 0. Sites with several indexes are given as `S -> {{1, "a"}, ...}`.
Apply the rules to an expanded Hamiltonian and compute its matrices with `makematricesbzvc` in a basis from `QSZaddspinket` (spins coupled to fermions) or with `makematricesbzop` in `spinbasis` (pure spin models; `spinclustermatrices` is faster for those).
An invalid site specification or a spin without a declared numeric size gives a message and returns `$Failed`.

Example. A single spin 1/2 without site index:

```
snegspinoperators[{S, 1/2}];
spinketbrarules[{S[]}]
```

spinketbraspinspin

`spinketbraspinspin[{s1, s2}]` gives the exchange operator $\mathbf{S}_1 \cdot \mathbf{S}_2$ of two spins *s1* and *s2* in the bra-ket representation, acting on the kets `ket[m1, m2]`.
The first spin occupies the first slot of the kets and the second spin the second slot, as in `spinbasis[{s1, s2}]`.
The result is expanded.

Example.

```
spinketbraspinspin[{1/2, 1/2}]
```

spinketbraX

`spinketbraX[s]` gives the x component of the spin-s operator in the bra-ket representation, $S^x = \sum_{mm'} |m\rangle (S^x)_{mm'} \langle m'|$.
`spinketbraX[s, pos]` places the spin in the slots of the kets and bras specified by the position array *pos*.
The result is a linear combination of products `nc[ket[m], bra[m']]` with the matrix elements of `spinmatrixX[s]`, $m, m' = s, \ldots, -s$. The analogous functions are `spinketbraY`, `spinketbraZ`, `spinketbraP` (S^+), `spinketbraM` (S^-), and `spinketbraI` (identity).
In this representation, spin operators act on the spin kets `ket[m]` by the rule $\langle m|m'\rangle = \delta_{mm'}$ (`braketrule`). The spin kets can be combined with fermion states in the occupation-number representation, `vc[..., ket[m]]`, which is used by NRG Ljubljana for impurity spins coupled to fermions.
The position array *pos* is a list of 0 and 1 of the length of the full kets; the spin is placed in the slot(s) with 1, and the other slots are `Null`. For two spins, `spinketbraX[s1, {1, 0}]` acts on the first slot of `ket[m1, m2]` and `spinketbraX[`

`s2, {0, 1}]` on the second; the product `nc` of operators on different slots combines them into operators on the full kets. The default is `{1}`.
An operator on a single slot does not act on full kets by itself; multiply it with the identities `spinketbraI` of the other slots. `spinketbrarules` does this automatically for abstract spin operators.
s must be an integer or a half-integer.

Example.
```
spinketbraX[1/2]
```

spinlessbasis

`spinlessbasis[bz]` projects the basis *bz* in the creation operator representation onto the states without spin-down electrons.
All creation operators with the spin index `D0` are set to 0; the states that vanish are removed, subspaces with equal quantum numbers are merged, and empty subspaces are dropped.
The quantum numbers are not changed: Q remains the charge relative to half filling of the spinful levels, $Q = N - N_{\text{levels}}$.
The result describes fully spin-polarized electrons, i.e. effectively spinless fermions represented by the spin-up operators.
bz must be in the creation operator representation (e.g. from `qbasis` or `qszbasis`).

Example.
```
snegfermionoperators[c];
spinlessbasis[qbasis[{c[1], c[2]}]]
```

SU2basistensorproduct

`SU2basistensorproduct[bz1, bz2, s1, s2, qn, fix1, fix2]` couples the SU(2) multiplets of the bases *bz1* and *bz2* to multiplets of the total spin with Clebsch-Gordan coefficients, and gives the basis of the highest-weight states of the coupled multiplets.
bz1 and *bz2* are bases `{{qn, states}, ...}` in the creation operator representation whose states are highest-weight states, $|S, S_z = S\rangle$, e.g. from `qsbasis` or `{First[ spinbasis[S]]}` (spin kets).
The functions in the arguments:
s1, *s2* give the spin of a subspace from its quantum numbers, e.g. `Last` for (Q, S), `First` for (S)
qn `qn[S, qn1, qn2]` gives the quantum numbers of the coupled states with total spin S; `qscombinefnc` gives $(Q_1 + Q_2, S)$, `qsspincombinefnc` gives (Q_1, S)

fix1, fix2 `fix[state, s, m]` gives the member with $S_z = m$ of the multiplet of the highest-weight *state* of spin *s*; `fixspin` lowers operator strings with `spindown`, `fixspinket` lowers spin kets `ket[m]`
For each pair of highest-weight states $|S_1 S_1\rangle$ and $|S_2 S_2\rangle$, the coupled highest-weight states are

$$|S, S\rangle = \sum_{m_1+m_2=S} \langle S_1 m_1; S_2 m_2 | S\, S\rangle\, |S_1 m_1\rangle |S_2 m_2\rangle, \quad S = |S_1 - S_2|, \ldots, S_1 + S_2,$$

with the coefficients of `ClebschGordan` (Condon-Shortley convention). The product $|S_1 m_1\rangle|S_2 m_2\rangle$ is `nc[fix1[state1, S1, m1], fix2[state2, S2, m2]]`.
The result is merged with `mergebasis`. Each state represents a multiplet of $2S + 1$ states.
The phases of the coupled states follow from the Clebsch-Gordan coefficients and from the order of the factors; they can differ from those of `qsbasis` for the same system.
`SU2basistensorproduct` does not call `makebasis`.

Example. Coupling the (Q, S) bases of two levels gives the (Q, S) basis of the dimer:

```
snegfermionoperators[c];
bz = SU2basistensorproduct[qsbasis[{c[1]}], qsbasis[{c[2]}], Last, Last,
     qscombinefnc, fixspin, fixspin]
```

szbasis

`szbasis[{c[1], c[2], ...}]` gives the basis of the Fock space of the levels `c[1]`, `c[2]`, ... in the creation operator representation, organized in subspaces with well-defined spin projection S_z.
`szbasis[ops]` is `bzvc2bzop[szbasisvc[ops]]`; see `szbasisvc` and `qszbasisvc`.
NRG Ljubljana uses `szbasis` for the symmetry types `SPU1` and `SPU1LR`.

Example.

```
snegfermionoperators[d];
szbasis[{d[]}]
```

szbasisvc

`szbasisvc[{c[1], c[2], ...}]` gives the basis of the Fock space of the levels `c[1]`, `c[2]`, ... in the occupation number representation, organized in subspaces with well-defined spin projection S_z.
`szbasisvc` constructs the (Q, S_z) basis with `qszbasisvc`, drops Q, and merges the subspaces with equal S_z. The result is `{{{Sz}, {states}}, ...}`.
The basis is appropriate for Hamiltonians that conserve S_z but not the particle number, e.g. with superconducting pairing and a magnetic field along z.

`szbasis` gives the same basis in the creation operator representation.

Example.

```
snegfermionoperators[d];
szbasisvc[{d[]}] // InputForm
```

transformbasis

`transformbasis[bz, op]` applies the operator *op* to all states of the basis *bz* in the occupation number representation, orthonormalizes the resulting states within each invariant subspace, and removes the empty subspaces.
bz is a basis `{{qn, states}, ...}` (see `makematricesbzvc`) with states in the occupation number representation, e.g. from `qszbasisvc` or `qsbasisvc`. The second argument is an operator expression.
For each subspace with orthonormal states $|i\rangle$, `transformfunc` computes $\hat{O}|i\rangle$, expands these vectors in the states $|i\rangle$, and orthonormalizes them (Gram-Schmidt, with vectors proven to vanish dropped). The result is an orthonormal basis of the image $\hat{O}\mathcal{H}_{qn}$ of the subspace, with the same quantum numbers.
The main application is a projector $\hat{P}$, which restricts the Hilbert space: `projector1[c[i]]` keeps the states with a singly occupied level *i* (local moment, Kondo and spin models), `projector0`, `projector2`, and `projector02` select empty, doubly occupied, or empty-or-double levels, and products of projectors combine the conditions.
The operator must conserve the quantum numbers of the basis. The components of $\hat{O}|i\rangle$ outside the subspace are discarded without warning.
The calculation uses exact arithmetic and `Simplify` and becomes slow for large subspaces.
The basis must be in the occupation number representation; convert with `bzop2bzvc`. For a basis in the creation operator representation, the evaluation does not terminate.

Example. Two-level basis with the first level `d` restricted to single occupancy:

```
snegfermionoperators[d, c];
bz = qsbasisvc[{d[], c[]}];
pb = transformbasis[bz, projector1[d[]]]
```

transformfunc

`transformfunc[states, op]` applies the operator *op* to the list of *states* in the occupation number representation and gives an orthonormal basis of the span of the results within the span of *states*.
`transformfunc` is the step of `transformbasis` for a single subspace. The states must be orthonormal. The vectors $\hat{O}|i\rangle$ are expanded in the *states* (components outside

their span are discarded) and orthonormalized with `orthogvc`; vectors that vanish are dropped.
The number of returned states is the rank of $\hat{O}$ on the span of *states*.

Example. Projection of the two-electron states with $S_z = 0$ of two levels onto single occupancy of both levels:

```
snegfermionoperators[c];
states = keyselect[qszbasisvc[{c[1], c[2]}], {0, 0}][[1, 2]]
```

transformQStoIS

`transformQStoIS[bz, Tm]` transforms the basis *bz* with the charge Q as the first quantum number, e.g. (Q, S), into a basis with the isospin I instead of Q, e.g. (I, S), using the isospin lowering operator *Tm*.
`transformQStoIS[bz]` uses the global value of `Tminus` as the isospin lowering operator.
The isospin (charge SU(2), pseudospin) of a spin-1/2 level i has the components $I_i^z = (n_i - 1)/2$, $I_i^+ = (-1)^{\phi_i} c_{i\uparrow}^\dagger c_{i\downarrow}^\dagger$, $I_i^- = (I_i^+)^\dagger$, with a phase $(-1)^{\phi_i}$ given by `isospinminus[c[i], phi]`. The total isospin $\mathbf{I} = \sum_i \mathbf{I}_i$ has $I^z = Q/2$, where $Q = N - N_{\text{levels}}$ is the charge of the basis.
For particle-hole symmetric Hamiltonians on bipartite lattices, e.g. the Hubbard model $-t \sum (c_{i\sigma}^\dagger c_{j\sigma} + \text{h.c.}) + \frac{U}{2} \sum_i (n_i - 1)^2$, the isospin is conserved if the phases alternate between the sublattices, $\phi_i = 0$ on one and $\phi_i = 1$ on the other. Together with the spin SU(2), this gives the SO(4) symmetry of the Hubbard model.
bz is a basis `{{qn, states}, ...}` (see `makematricesbzvc`) in the occupation number representation whose first quantum number is Q; the further quantum numbers (e.g. S or S_z) are kept. Suitable bases are `qsbasisvc`, `qszbasisvc`, and `qbasisvc`.
The result contains only the highest-weight states of the isospin multiplets, $I^z = I$, as the (Q, S) basis contains only the states with $S_z = S$. The first quantum number is I; each state represents a multiplet of $2I + 1$ states with $Q = 2I, 2I - 2, \ldots, -2I$. In an (I, S) basis, an eigenvalue in the subspace (I, S) is $(2I + 1)(2S + 1)$-fold degenerate.
The subspaces are processed in the order of decreasing Q: the states with the largest Q are highest-weight states; I^- generates the other members of their multiplets, which are projected out of the subspaces with lower Q. The remaining states are the highest-weight states of the next multiplets. Empty subspaces are removed; the result is ordered by decreasing $Q = 2I$ of the original subspaces.
Tm must be the total isospin lowering operator of all levels in the basis, e.g. `isospinminus[c[1], 0] + isospinminus[c[2], 1]`.
If `Tminus` has no value, `transformQStoIS[bz]` issues the message `transformQStoIS::Tminus` and returns `$Failed`.

`quickISObasis`, `quickISOSZbasis`, and `quickSU2basis` construct the lowering operator with the phases given by `nnop` and call `transformQStoIS`.

Example. The (I, S) basis of a single level: the empty and doubly occupied states form an isospin doublet ($I = 1/2$, $S = 0$), the singly occupied states a spin doublet ($I = 0$, $S = 1/2$):

```
snegfermionoperators[c];
transformQStoIS[qsbasisvc[{c[1]}], isospinminus[c[1]]]
```

transformtoLR

`transformtoLR[bz, map]` transforms the basis *bz* in the creation operator representation into a basis with an additional quantum number, the parity $P = \pm 1$ under the mirror reflection defined by the list of levels *map*.

`transformtoLR[bz, map, rules]` applies the additional substitution *rules* together with the reflection.

bz is a basis `{{qn, states}, ...}` (see `makematricesbzvc`) whose states are operator expressions, as generated by `qsbasis`, `qszbasis`, or `bzvc2bzop`. The result has the same form, with the parity appended to each list of quantum numbers, e.g. $(Q, S) \to (Q, S, P)$.

The reflection $\hat{P}$ is defined by `lrmap[map]`: the levels in *map* are mapped onto the levels at the mirror positions, `map[[k]]` $\leftrightarrow$ `map[[-k]]`. For a state $f(c^\dagger)|0\rangle$, $\hat{P}f(c^\dagger)|0\rangle$ is obtained by substituting the creation operators in f; the vacuum is even. Fermionic signs from reordering the operators are taken into account.

For each subspace with states $|i\rangle$, the even and odd combinations $(|i\rangle \pm \hat{P}|i\rangle)/2$ are orthonormalized within the span of the subspace (Gram-Schmidt); the resulting states have $P = +1$ and $P = -1$. Empty subspaces are removed, and the result is sorted by the quantum numbers.

Each subspace of *bz* must be mapped onto itself by the reflection. This holds for the charge Q, the spin S and S_z, and the isospin, provided all levels exchanged by the reflection are in the basis.

The optional *rules* describe the transformation of further degrees of freedom under the reflection, e.g. `{ket[k_] :> (-1)^k ket[k]}` for a phonon mode that is odd under reflection, or a sign change of orbitals with odd parity.

In a system with mirror symmetry, the Hamiltonian commutes with $\hat{P}$ and is block-diagonal in the parity basis. For an impurity coupled symmetrically to two leads L and R, the even combination $(c_L + c_R)/\sqrt{2}$ couples to the impurity and the odd combination $(c_L - c_R)/\sqrt{2}$ decouples; the parity basis separates the corresponding subspaces.

The calculation is done in the creation operator representation and is slow beyond two sites (about 15 s for three levels in a (Q, S) basis). `transformtoLRvc` performs

the same transformation and returns the basis in the occupation number representation in a fraction of this time.

Example. Basis of the Hubbard dimer with quantum numbers (Q, S, P):

```
snegfermionoperators[c];
snegrealconstants[t, U];
lr = transformtoLR[qsbasis[{c[1], c[2]}], {c[1], c[2]}]
```

transformtoLRvc

`transformtoLRvc[bz, map, vac]` transforms the basis *bz* in the creation operator representation into a basis with the additional parity quantum number $P = \pm 1$, with the states in the occupation number representation obtained by acting on the vector *vac*.
`transformtoLRvc[bz, map, vac, rules]` applies the additional substitution *rules* together with the reflection.
The parity and the conventions are those of `transformtoLR`: the reflection is defined by `lrmap[map]`, and the parity is appended to the quantum numbers of each subspace.
bz must be in the creation operator representation, e.g. from `qsbasis` or `qszbasis`.
vac is normally `vacuum[]`, the empty occupation number vector of the current `BASIS`.
The reflected states are formed by substitution in the operator expressions, converted with `bzop2bzvc`, and orthonormalized in the occupation number representation, which is much faster than `transformtoLR`.
The result equals `bzop2bzvc[transformtoLR[bz, map, rules], vac]` and is used with `makematricesbzvc`.

Example. Parity-adapted basis of the Hubbard dimer in the occupation number representation:

```
snegfermionoperators[c];
lr = transformtoLRvc[qsbasis[{c[1], c[2]}], {c[1], c[2]}, vacuum[]]
```

transformtoPH

`transformtoPH[bz, {nph}]` gives the product of the basis *bz* with the states $|n\rangle$, $n = 0, \ldots, n_{\mathrm{ph}}$, of a local phonon mode.
`transformtoPH[bz, {n1, n2, ...}]` includes several phonon modes with the cutoffs $n_1, n_2, \ldots$; the phonon states are `ket[k1, k2, ...]`.
`PH` stands for phonons. The phonon (boson) states are represented by kets: `ket[n]` is $|n\rangle = (b^\dagger)^n|0\rangle/\sqrt{n!}$, truncated at n_{ph} quanta.
bz is a basis `{{qn, states}, ...}` (see `makematricesbzvc`) in either representation.
Each state of each subspace is multiplied by all phonon states, and the quantum

numbers are not changed; the dimension of every subspace grows by the factor $\prod_i (n_i + 1)$. In the creation operator representation, the states are `nc[f, ket[n]]`; in the occupation number representation, `vc[..., ket[n]]`.
Within a subspace, the phonon state is the outer index: first all fermion states with $|0\rangle$, then all with $|1\rangle$, and so on.
The phonon operators in the truncated space are `phononnumber[nph]` ($b^\dagger b$), `phononplus[nph]` ($b^\dagger$), `phononminus[nph]` (b), and `phononx[nph]` ($b + b^\dagger$), written with `ket` and `bra`; for several modes, `phononplus[i, {n1, n2, ...}]` etc. act on mode i.
The truncation is exact for operators that conserve the phonon number. With electron-phonon coupling, results converge with increasing cutoff; check the convergence. The truncated operators violate $[b, b^\dagger] = 1$ in the highest state.
The cutoffs must be given as a list of integers; otherwise `transformtoPH` stays unevaluated.
To combine phonons with the parity of `transformtoLR`, add the phonons first and give the transformation of the kets under the reflection in the rules of `transformtoLR`.

Example. Single level with a phonon mode with at most two quanta:

```
snegfermionoperators[d];
transformtoPH[qsbasis[{d[]}], {2}]
```

vacuum

`vacuum[]` gives the vacuum state $|0\rangle$ in the occupation number representation, `vc[0, 0, ..., 0]`, for the single-particle basis `BASIS` defined by `makebasis`.
The length of the vector is the number of single-particle states in `BASIS`: twice the number of spin-1/2 levels.
`vacuum[]` is evaluated with the current `BASIS`; call it after `makebasis` or after a basis generator such as `qszbasisvc`, which calls `makebasis`.
`ap[1, VACUUM]` and `ap[op, VACUUM]` also give `vacuum[]` and `ap[op, vacuum[]]`: the symbol `VACUUM` (the vacuum in the creation operator representation) is replaced by `vacuum[]` when an operator is applied with `ap`.
If `BASIS` is empty, `vacuum[]` is `vc[]`.
The old usage message refers to the help page `vacuumlc`; the function is `vacuum`.

Example.

```
snegfermionoperators[c];
makebasis[{c[1], c[2]}];
vacuum[] // InputForm
```

vc

`vc[n1, n2, ..., nN]` represents the Fock state with occupation numbers $n_k \in \{0, 1\}$ of the single-particle states in the list `BASIS` defined by `makebasis`.

`vc[n1, ..., nN, ket[...]]` is the product of a fermionic Fock state with a Dirac ket of additional degrees of freedom (spins, phonons).

`makebasis[{c[1], c[2], ...}]` sets the global list `BASIS` of single-particle creation operators. For spin-1/2 levels the order is $c^\dagger_{1\uparrow}, c^\dagger_{1\downarrow}, c^\dagger_{2\uparrow}, c^\dagger_{2\downarrow}, \ldots$; `op2ndx` and `ndx2op` convert between operators and positions.

Sign convention: with $a^\dagger_k$ the k-th element of `BASIS`,

$$\mathrm{vc}[n_1, n_2, \ldots, n_N] = (a^\dagger_1)^{n_1} (a^\dagger_2)^{n_2} \cdots (a^\dagger_N)^{n_N} |0\rangle,$$

i.e., the creation operators act in the order of `BASIS`, with the operator of the lowest index on the left. Applying $a^\dagger_k$ or a_k to this state gives the fermionic sign $(-1)^{\sum_{l<k} n_l}$ (see `ap`).

`vc` states form an orthonormal basis: `scalarproductvc[vc[...], vc[...]]` is 1 for identical vectors and 0 otherwise. General states are linear combinations `x vc[...] + y vc[...]` with scalar coefficients.

Operators act on `vc` states with `ap[op, v]`. In a product `nc[a, b, ..., v]` that ends with a `vc` state, `ap` is called automatically.

`vacuum[]` is `vc[0, 0, ..., 0]` with the length of `BASIS`. `vc2ops` converts a `vc` state into the equivalent string of creation operators (the creation operator representation); `ops2vc` converts back.

A `ket` as the last element describes non-fermionic degrees of freedom. Kets are appended by `ap[ket[...], v]`, bras act on them by `braketrule`, and `scalarproductvc` multiplies the fermionic overlap with the braket of the kets.

`vc` has the attributes `NHoldAll` and `Protected`: `N` does not change the occupation numbers.

With `PrettyOutput` (the default), `vc` states are displayed as rows of filled (occupied) and empty (unoccupied) rectangles, red for spin up and blue for spin down; see `vccolor`.

The `vc` vectors are meaningful only together with the `BASIS` for which they were constructed. Basis generators such as `qszbasisvc` and `qsbasisvc` call `makebasis` and thereby redefine `BASIS`.

Example. Single-particle basis of one spin-1/2 level `d[]`:

```
snegfermionoperators[d];
makebasis[{d[]}]
```

vc2ops

vc2ops[expr] converts the states vc[...] in *expr* from the occupation number representation into creation operator strings acting on the vacuum.

vc[n1, n2, ...] is converted into nc[a1, a2, ...] of the creation operators $a_k^\dagger$ of BASIS with $n_k = 1$, in the order of BASIS: vc$[n_1, n_2, ...] \to (a_1^\dagger)^{n_1}(a_2^\dagger)^{n_2}\cdots$. The product is then brought to the normal order of nc, which can change the sign.

The vacuum vacuum[] is converted into 1.

A trailing ket is kept: vc[n1, ..., ket[k]] gives nc[..., ket[k]].

vc2ops is linear and Listable. bzvc2bzop converts a whole basis.

If the length of the vector differs from the length of BASIS, the message vc2ops::incom is issued and Infinity is returned.

The inverse conversion is ops2vc (or ap[ops, vacuum[]]): ops2vc[vc2ops[v]] is v.

Example.

```
snegfermionoperators[c];
makebasis[{c[1], c[2]}];
vc2ops[vc[1, 0, 0, 1]]
```

vccolor

vccolor is the function used to display the occupation numbers of vc vectors in StandardForm.

With PrettyOutput (the default), vc[n1, n2, ...] is displayed as $\|...\rangle$ with the elements returned by vccolor[{n1, n2, ...}] in between.

The default is vccolorDefault, which shows occupied states as filled and empty states as open rectangles, alternately red (spin up) and blue (spin down), as appropriate for spin-1/2 levels. vccolorSimple shows black rectangles without spin coloring, which is preferable for spinless fermions and fermions with other spins.

Set vccolor = vccolorSimple to change the display, and vccolor = vccolorDefault to restore it. Any function that maps the list of entries to a list of displayable elements can be used.

The formatting does not affect InputForm or the internal representation.

Example.

```
snegfermionoperators[c];
makebasis[{c[1], c[2]}];
v = vc[1, 0, 1, 1]
```

vccolorDefault

`vccolorDefault[{n1, n2, ...}]` gives the default display elements of the occupation numbers of a `vc` vector: filled (occupied) and open (empty) rectangles, red for odd positions (spin up) and blue for even positions (spin down).
`vccolorDefault` is the default value of `vccolor`. It assumes that the entries come in pairs $(n_{i\uparrow}, n_{i\downarrow})$, as for spin-1/2 levels; for an odd number of entries, the last entry is taken to be a ket and is shown unchanged.
For spinless fermions or fermions with other spins, the coloring does not correspond to the spin; use `vccolorSimple`.

Example.

```
vccolorDefault[{1, 0, 0, 1}]
```

vccolorSimple

`vccolorSimple[{n1, n2, ...}]` gives display elements of the occupation numbers of a `vc` vector without spin coloring: filled rectangles for occupied and open rectangles for empty states.
Set `vccolor = vccolorSimple` to use this display for all `vc` vectors; `vccolor = vccolorDefault` restores the default.

Example.

```
vccolorSimple[{1, 0, 0, 1}]
```

zeroonvac

`zeroonvac[expr]` drops the terms of the operator expression *expr* that annihilate the vacuum, i.e., the terms whose rightmost operator is an annihilation operator.
`zeroonvac` simplifies states in the creation-operator representation, $|\psi\rangle = expr\,|0\rangle$, where the vacuum $|0\rangle$ is not written. It is used, e.g., after applying an operator to a state: `zeroonvac[nc[op, state]]`.
A term is dropped if its last operator annihilates the vacuum, as decided by `isannihilation`; the vacuum is the one set by `ordering`. For `ordering[c] = SEA`, c_k with $k > 0$ and $c_k^\dagger$ with $k < 0$ annihilate the Fermi sea.
The result relies on the normal order of the products, which `nc` establishes automatically.
`zeroonvac` is linear and `Listable`. If the term ends with a `ket`, the operators to the left of the ket are examined.
If the sign of a symbolic index is undecided for `ordering[c] = SEA`, the result contains an unevaluated `If[...]`. `snegAssuming` can be used to specify the sign.

Symbolic sums `sum[a, {k}]` are not processed; the variant `Zeroonvac` also applies `zeroonvac` to the summands.

Example.

```
snegfermionoperators[c];
zeroonvac[c[CR, 1, UP] + nc[c[CR, 1, UP], c[AN, 2, UP]] + c[AN, 1, UP]]
```

15.7 Expectation Values and Wick's Theorem

braketop

`braketop[a, op, b]` gives the matrix element $\langle a|\hat{O}|b\rangle$, where the states are given in the creation operator representation, $|a\rangle = \hat{A}|0\rangle$ and $|b\rangle = \hat{B}|0\rangle$.
`braketop[a, op, b]` computes the vacuum expectation value $\langle 0|\hat{A}^\dagger\hat{O}\hat{B}|0\rangle$ with `vev`. a and b are operator expressions, usually strings of creation operators; 1 represents the vacuum.
`braketop` does not require `makebasis`; it works for any operators known to `vev`.
`braketop` is `Listable`. `expvop[op, a]` is `braketop[a, op, a]`. For states in the occupation number representation, use `braketvc`, which is faster.

Example.

```
snegfermionoperators[c];
braketop[c[CR, 2, UP], hop[c[1], c[2]], c[CR, 1, UP]]
```

contraction

`contraction[a, b]` gives the contraction of the operators a and b, $\langle ab\rangle = ab - :ab:$, in this order.
The contraction is the difference between the product and its normal-ordered form with respect to the vacuum set by `ordering`. It is a c-number and equals the vacuum expectation value $\langle 0|ab|0\rangle$.
For fermions with the empty vacuum (`ordering[c] = EMPTY`, the default), $\langle c_\alpha c^\dagger_\beta\rangle = \delta_{\alpha\beta}$ and all other contractions vanish. For bosons, $\langle a_\alpha a^\dagger_\beta\rangle = \delta_{\alpha\beta}$.
For the Fermi sea (`ordering[c] = SEA`), $\langle c^\dagger_{k\sigma}c_{k'\sigma'}\rangle = \delta_{kk'}\delta_{\sigma\sigma'}\theta(-k)$ and $\langle c_{k\sigma}c^\dagger_{k'\sigma'}\rangle = \delta_{kk'}\delta_{\sigma\sigma'}(1-\theta(-k))$, where k is the second index (the energy measured from the Fermi level) and θ is `UnitStep`.
For operators with `ordering[c] = NONE`, the contraction of an annihilation operator with a creation operator to its right is the canonical (anti)commutator `acmt` or `cmt`, and all other contractions are zero.
Operators of different families have zero contraction.

The arguments must be single operators. The order of the arguments matters: `contraction[a, b]` is the contraction with a to the left of b.

Example. Contractions for the empty vacuum:

```
snegfermionoperators[c];
snegfreeindexes[i, j];
{contraction[c[AN, i, UP], c[CR, j, UP]], contraction[c[CR, i, UP], c[AN
    , j, UP]]}
```

dd

`dd[expr]` represents the normal-ordered product $:expr:$.
`dd` is a formal placeholder for normal-ordered products. It is returned by `wick`; the operators inside are kept in the order produced by `nc`. To compute an explicit normal-ordered operator, use `normalorder`, `normalorderwick`, or `wick`.
The normal order refers to the vacuum set by `ordering` (empty state or Fermi sea). By definition, `vev[dd[x]]` and `vevwick[dd[x]]` are 0.
`dd` is linear: it distributes over sums and takes numbers and constants out. Nested `dd` are flattened, and `dd[]` is 1.
In output, `dd[x]` is shown with green double dots, $:x:$.

Example. `wick` writes products in terms of normal-ordered products and contractions:

```
snegfermionoperators[c];
ordering[c] = SEA;
snegfreeindexes[k1, k2];
wick[nc[c[AN, k1, UP], c[CR, k2, UP]]]
```

expvop

`expvop[op, a]` gives $\langle a|op|a\rangle$ for the operator op and the state $|a\rangle = a|0\rangle$ in the creation-operator representation.
`expvop[op, a]` is `braketop[a, op, a]`, i.e., `vev[nc[conj[a], op, a]]`. The vacuum is the one set by `ordering`.
The state is not normalized: for a normalized expectation value, divide by `scalarproductop[a, a]` or use a normalized state.
`expvop` is `Listable` in both arguments.

Example. Total spin $\mathbf{S}^2$ of the normalized singlet and triplet states of two electrons:

```
snegfermionoperators[c];
s = (nc[c[CR, 1, UP], c[CR, 2, DO]] - nc[c[CR, 1, DO], c[CR, 2, UP]])/
    Sqrt[2];
```

```
t0 = (nc[c[CR, 1, UP], c[CR, 2, DO]] + nc[c[CR, 1, DO], c[CR, 2, UP]])/
    Sqrt[2];
stot = spinss[c[1]] + spinss[c[2]] + 2 spinspin[c[1], c[2]];
{expvop[stot, s], expvop[stot, t0]}
```

normalorderwick

`normalorderwick[expr]` gives `expr - vevwick[expr]`, the operator *expr* with its vacuum expectation value, computed by Wick's theorem, subtracted.
`normalorderwick` is `normalorder` with `vevwick` instead of `vev`. It is needed for the Fermi sea (`ordering[c] = SEA`), where `vev` does not evaluate products of more than two operators with symbolic indexes.
For a product of two operators, the result is the normal-ordered product $:AB: = AB - \langle AB \rangle$. For longer products, $x - \langle x \rangle$ differs from $:x:$ by the terms with partial contractions; `wick` gives the full expansion.
`normalorderwick` is `Listable`.

Example. A density-density product in the Fermi sea with its expectation value subtracted:

```
snegfermionoperators[c];
ordering[c] = SEA;
snegfreeindexes[k1, k2];
x = nc[c[CR, k1, UP], c[AN, k1, UP], c[CR, k2, DO], c[AN, k2, DO]];
normalorderwick[x]
```

vev

`vev[expr]` gives the vacuum expectation value $\langle 0|expr|0\rangle$ of the operator expression *expr*.
The vacuum $|0\rangle$ is the reference state of the operator family, set by `ordering`: the empty state (no particles) for `ordering[c] = EMPTY` (the default) and the Fermi sea for `ordering[c] = SEA`. For bosons, the vacuum is the state without bosons.
`vev` is linear: it distributes over sums, and numbers, declared constants, and Grassmann numbers are taken out. `vev[x] = x` for scalars x. `vev[sum[a, {k}]]` is `sum[vev[a], {k}]`.
A product `nc[o1, ..., on]` has zero expectation value if its last factor annihilates the vacuum (`isannihilation`) or its first factor annihilates the bra $\langle 0|$ (`iscreation`).
A single fermion or boson operator and a product of an odd number of fermion or boson operators have zero expectation value: the vacuum has a definite particle number.

For `ordering[c] = SEA`, the bilinears are $\langle c^\dagger_{k\sigma} c_{k'\sigma'}\rangle = \delta_{kk'}\delta_{\sigma\sigma'}\,\theta(-k)$ and $\langle c_{k\sigma} c^\dagger_{k'\sigma'}\rangle = \delta_{kk'}\delta_{\sigma\sigma'}\,\theta(k)$, where θ is `UnitStep` with $\theta(0) = 1$.
`vev[dd[...]] = 0`: the expectation value of a normal-ordered product (see `dd`) vanishes by definition. `vev[vev[x]] = vev[x]`.
For `ordering[c] = SEA`, the second index of `c[t, k, ..., σ]` (the first after `CR/AN`) is the single-particle energy measured from the Fermi level: levels with $k \le 0$ are occupied in the Fermi sea, levels with $k > 0$ are empty. `iscreation` and `isannihilation` evaluate the sign of k; for symbolic k they remain undecided and the corresponding rules do not apply.
For `ordering[c] = EMPTY`, `nc` brings every product to normal order, so `vev` only has to pick the terms without operators. For the Fermi sea, products of more than two operators with symbolic indexes are generally not evaluated by `vev`; use `vevwick`, which applies Wick's theorem.
With the state placeholder `VACUUM`, `nc[conj[VACUUM], VACUUM]` is 1, so matrix elements can also be written out explicitly.
`vev` is `Listable`. Expressions with undeclared symbols (treated as operators of unknown type) are returned unevaluated.

Example. Expectation values in the empty vacuum, $\langle 0|c_{1\uparrow}c^\dagger_{1\uparrow}|0\rangle = 1$ and $\langle 0|c^\dagger_{1\uparrow}c_{1\uparrow}|0\rangle = 0$:

```
snegfermionoperators[c];
{vev[nc[c[AN, 1, UP], c[CR, 1, UP]]], vev[nc[c[CR, 1, UP], c[AN, 1, UP
    ]]]}
```

vevwick

`vevwick[expr]` gives the vacuum expectation value of the operator expression *expr*, computed with Wick's theorem.
Wick's theorem expresses the expectation value of a product of creation and annihilation operators in a vacuum (Slater determinant or boson vacuum) as the sum over all complete pairings of the operators into contractions,

$$\langle A_1 A_2 \cdots A_{2n}\rangle = \sum_{\text{pairings}} (\pm 1)^P \langle A_{i_1} A_{j_1}\rangle\langle A_{i_2} A_{j_2}\rangle \cdots \langle A_{i_n} A_{j_n}\rangle, \qquad i_m < j_m,$$

where $(-1)^P$ is the sign of the permutation that brings the paired fermion operators next to each other (no sign for bosons).
The contractions are those of `contraction`: $\langle AB\rangle = AB - {:}AB{:}$ for the vacuum set by `ordering`. For the empty vacuum, $\langle c_i c^\dagger_j\rangle = \delta_{ij}$ and $\langle c^\dagger_i c_j\rangle = 0$; for the Fermi sea (`ordering[c] = SEA`), $\langle c^\dagger_{k\sigma} c_{k'\sigma'}\rangle = \delta_{kk'}\delta_{\sigma\sigma'}\theta(-k) = \delta_{kk'}\delta_{\sigma\sigma'} n_k$ and $\langle c_{k\sigma} c^\dagger_{k'\sigma'}\rangle = \delta_{kk'}\delta_{\sigma\sigma'}(1 - n_k)$, where the second index is the single-particle energy measured from the Fermi level.

Only pairs of one operator with the first index `CR` and one with `AN` are contracted. Anomalous contractions $\langle cc\rangle$, $\langle c^\dagger c^\dagger\rangle$ are zero in these vacua.
`vevwick` acts on the product after the automatic reordering by `nc`. It is linear: sums are expanded, constants are taken out, and `vevwick[sum[a, {k}]]` is `sum[vevwick[a], {k}]`. Single operators, products of an odd number of fermion operators, and normal-ordered products `dd[...]` give 0.
The result contains `KroneckerDelta` and `UnitStep` factors. `SimplifyKD` simplifies products of them, and `sumSimplifyKD` performs sums over Kronecker deltas. `SimplifyKD` replaces $1-\theta(-k)$ by $\theta(k)$, which is valid for $k\neq 0$; the level $k=0$ counts as occupied ($\theta(0)=1$) in `vev` and `vevwick`.
For the empty vacuum, `nc` already brings products to normal order, and `vevwick` gives the same result as `vev`. `vevwick` is needed for the Fermi sea, for products of operators with `ordering[c] = NONE`, and for products that `vev` does not reduce.
The number of complete pairings of n creation and n annihilation operators is $n!$. Products of six operators take about 0.01 s, products of eight operators about 0.1 s. `vevwick` is `Listable`.

Example. Expectation values in the Fermi sea. The second index is the energy measured from the Fermi level, and `UnitStep[-k]` is the occupation n_k:

```
snegfermionoperators[c];
ordering[c] = SEA;
snegfreeindexes[k1, k2, k3, k4];
vevwick[nc[c[CR, k1, UP], c[AN, k2, UP]]]
```

wick

`wick[expr]` rewrites the operator products in *expr* as sums of normal-ordered products `dd[...]` multiplied by contractions (Wick's theorem).
Wick's theorem for a product of operators $A_1\cdots A_m$ reads

$$A_1A_2\cdots A_m = :A_1\cdots A_m: + \sum_{(ij)} \pm\langle A_iA_j\rangle :\cdots: + \sum_{(ij)(kl)} \pm\langle A_iA_j\rangle\langle A_kA_l\rangle :\cdots: + \cdots,$$

where the sums run over all sets of one, two, ... pairs $(i<j)$, the normal-ordered products $:\cdots:$ contain the remaining operators in their original order, and the sign is the parity of the permutation of the fermion operators.
The normal-ordered products are represented by `dd[...]`, and the contractions $\langle A_iA_j\rangle$ are computed with `contraction`. The normal order refers to the vacuum set by `ordering`: the empty state (`EMPTY`) or the Fermi sea (`SEA`).
`wick` acts on the product after the automatic reordering by `nc`. For the empty vacuum, `nc` already gives normal order, so the expansion contains no contractions; `wick` is mainly useful for the Fermi sea and for operators with `ordering[c] = NONE`. Only pairs of one operator with the first index `CR` and one with `AN` are contracted.

`wick` is linear and `Listable`, and `wick[sum[a, {k}]]` is `sum[wick[a], {k}]`. `wick[1]` is 1.
Dropping all terms with `dd[...]` gives the expectation value; this is what `vevwick` computes.
The number of terms grows factorially with the length of the product.

Example. A bilinear in the Fermi sea is its normal-ordered part plus the contraction:

```
snegfermionoperators[c];
ordering[c] = SEA;
snegfreeindexes[k1, k2, k3, k4];
wick[nc[c[CR, k1, UP], c[AN, k2, UP]]]
```

15.8 Quadratic Hamiltonians and Mean-Field Theory

avg

`avg[x]` represents the expectation value $\langle x \rangle$ of the operator expression x.
`avg` is the symbolic expectation value used by `meanfield`, `meanfieldsolve`, `spectralmoments`, and `greensfunction`. It has no definite value; numerical values are obtained with `meanfieldsolve`, or by replacing the averages with the results of `gaussianaverage` or `thermalaverage`.
`avg` is linear: c-numbers are taken out, $\langle ax + by \rangle = a\langle x \rangle + b\langle y \rangle$, and the average of a c-number is the c-number.
The argument is the canonically ordered operator expression, as produced by `nc`. Therefore $\langle cc^\dagger \rangle$ evaluates to $1 - \langle c^\dagger c \rangle$ for a fermion, and different orderings of the same product are identified.
The average of a single fermion operator is 0. Averages of other odd products are kept.
Complex conjugation gives the average of the Hermitian conjugate: $\langle x \rangle^* = \langle x^\dagger \rangle$.
`avg[...]` is treated as a number in operator products (`isnumericQ` gives `True`), so mean-field Hamiltonians are quadratic operators for `nc`, `quadraticform`, and `bdghamiltonian`.
Averages of symbolic sums are sums of averages: `avg[sum[x, {k}]]` is `sum[avg[x], {k}]`.
In `StandardForm`, `avg[x]` is displayed as $\langle x \rangle$.

Example. The average is linear and acts on the canonically ordered expression:

```
snegfermionoperators[c];
{avg[3 number[c[1], UP] + 2], avg[nc[c[AN, 1, UP], c[CR, 1, UP]]]}
```

bdghamiltonian

`bdghamiltonian[H, {ck[a], ck[b], ...}, k]` returns `{Hk, E0}`, the Bloch Bogoliubov-de Gennes matrix $H(k)$ and the constant E_0 with $H = \frac{1}{2}\sum_k \Psi_k^\dagger H(k) \Psi_k + E_0$, for a translation-invariant quadratic Hamiltonian H in momentum space, with the Nambu spinor $\Psi_k = (c_{ka}, c_{kb}, \dots, c_{-ka}^\dagger, c_{-kb}^\dagger, \dots)^T$.
H is a symbolic sum over the momentum index k (declared with `snegmomentumindexes`), typically obtained with `fourier` from a lattice Hamiltonian. It may contain anomalous terms $c_k^\dagger c_{-k}^\dagger$, $c_{-k}c_k$ (fermions) or $b_k^\dagger b_{-k}^\dagger$, $b_{-k}b_k$ (bosons), which `blochhamiltonian` does not describe.
The orbitals are given as templates, as in `blochhamiltonian`: `ck` or `ck[a...]`. The momentum is inserted at the position given by `"IndexPosition"` (by default as the first index after `CR/AN`); for fermions with spin, the spin index is added as the last index and varies fastest ($\uparrow, \downarrow$ for spin 1/2, `0` for spinless fermions). For example, `{ck}` with spin-1/2 fermions gives $\Psi_k = (c_{k\uparrow}, c_{k\downarrow}, c_{-k\uparrow}^\dagger, c_{-k\downarrow}^\dagger)$.
For n orbital components, $H(k)$ is a $2n \times 2n$ matrix. For fermions, its blocks are $h(k)$ (upper left), $\Delta(k)$ (upper right), $\Delta(k)^\dagger$ (lower left), and $-h(-k)^T$ (lower right), where $h(k)$ is the Bloch Hamiltonian of the normal part. For bosons, the lower right block is $+h(-k)^T$.
The matrix elements are $H_{pq}(k) = \eta_p \eta_q [\Psi_{k,p}, [H, \Psi_{k,q}^\dagger]]_\mp$ (anticommutator for fermions, commutator for bosons, $\eta = 1$ for fermions and $\eta = (1, \dots, 1, -1, \dots, -1)$ for bosons), evaluated with the lattice sums performed by `sumSimplifyKD`.
The constant is $E_0 = C + \frac{1}{2}\sum_k \operatorname{Tr} h(k)$ for fermions and $E_0 = C - \frac{1}{2}\sum_k \operatorname{Tr} h(k)$ for bosons, where C is the c-number part of the normal-ordered H. Since Ψ_k and Ψ_{-k} contain the same operators, every mode appears twice in the sum over k, which accounts for the factor $\frac{1}{2}$.
In contrast to the real-space BdG matrix of `quadraticform`, $H(k)$ relates k and $-k$: the particle-hole symmetry reads $\tau_x H(k)^* \tau_x = -H(-k)$ (fermions) or $H(-k)$ (bosons). To diagonalize $H(k)$ use `bogoliubov[Hk, "Fermion", "Momentum" -> True]` or `"Boson"`; averages are computed with `gaussianaverage[..., "Momentum" -> k]`.
Functions of k such as `ε[k]` stay as given; $h(-k)$ contains `ε[-k]`. For inversion-symmetric dispersions replace `ε[-k] -> ε[k]` before diagonalization.
The following options can be given:
`"IndexPosition" 1` position of the momentum among the indexes after `CR/AN`
`"Check" True` check translation invariance and Hermiticity
With `"Check" -> True`, terms that do not conserve momentum give the message `bdghamiltonian::transl` and `$Failed`; a non-Hermitian $H(k)$ gives the warning `bdghamiltonian::herm`.
Other failures: k is not a declared momentum index (`bdghamiltonian::index`), H

contains terms with an odd number of mode operators (`bdghamiltonian::linear`), terms of higher order or other operators (`bdghamiltonian::rest`), modes of mixed statistics (`bdghamiltonian::stat`), or fermions declared with an ordering other than the default `EMPTY` (`bdghamiltonian::ordering`), for which E_0 cannot be determined.

Example. The Kitaev chain, $H = \sum_R [-t(f_R^\dagger f_{R+1} + \text{h.c.}) - \mu n_R + \Delta(f_R^\dagger f_{R+1}^\dagger + f_{R+1} f_R)]$, in momentum space:

```
snegspinlessfermionoperators[f, fk];
snegrealconstants[t, mu, Dl];
sneglatticeindexes[{R}, Ns];
snegmomentumindexes[{k}, Ns];
H = sum[-t hop[f[R], f[R + 1]] - mu number[f[R]] + Dl (nc[f[CR, R, 0], f
    [CR, R + 1, 0]] + nc[f[AN, R + 1, 0], f[AN, R, 0]]), {R}];
Hk = fourier[H, f -> fk, R -> k];
{hk, E0} = bdghamiltonian[Hk, {fk}, k]
```

bogoliubov

`bogoliubov[M, "Fermion"]` diagonalizes the fermionic Bogoliubov-de Gennes (BdG) matrix M and returns `{ε, T}`, the quasiparticle energies ε_m and the transformation matrix T with $\Psi = T\Phi$.

`bogoliubov[M, "Boson"]` does the same for a bosonic BdG matrix (paraunitary transformation).

`bogoliubov[H, {a1, a2, ...}]` diagonalizes the quadratic Hamiltonian H of the modes a_i and returns `{ε, T, Egs}`, including the ground-state energy E_{gs}.

`bogoliubov[Hk, type, "Momentum" -> True]` diagonalizes a Bloch BdG matrix $H(k)$ from `bdghamiltonian`.

The BdG matrix refers to $H = \frac{1}{2}\Psi^\dagger M \Psi + E_0$ with $\Psi = (a_1, \ldots, a_n, a_1^\dagger, \ldots, a_n^\dagger)^T$, as returned by `quadraticform`. The quasiparticle spinor is $\Phi = (\gamma_1, \ldots, \gamma_n, \gamma_1^\dagger, \ldots, \gamma_n^\dagger)^T$, and T has the blocks U (upper left), V^* (upper right), V (lower left), and U^* (lower right), i.e. $a_i = \sum_m (U_{im}\gamma_m + V_{im}^*\gamma_m^\dagger)$. Column m of T contains the coefficients $(U_{1m}, \ldots, U_{nm}, V_{1m}, \ldots, V_{nm})$ of the quasiparticle γ_m.

Fermions: T is unitary and $T^\dagger M T = \text{diag}(\varepsilon_1, \ldots, \varepsilon_n, -\varepsilon_1, \ldots, -\varepsilon_n)$. Then $H = \sum_m \varepsilon_m \gamma_m^\dagger \gamma_m + E_{\text{gs}}$ with $E_{\text{gs}} = E_0 - \frac{1}{2}\sum_m \varepsilon_m$.

Bosons: T is paraunitary, $T^\dagger \eta T = \eta$ with $\eta = \text{diag}(1, \ldots, 1, -1, \ldots, -1)$, which preserves the commutation relations, and $T^\dagger M T = \text{diag}(\varepsilon_1, \ldots, \varepsilon_n, \varepsilon_1, \ldots, \varepsilon_n)$. Then $H = \sum_m \varepsilon_m \gamma_m^\dagger \gamma_m + E_{\text{gs}}$ with $E_{\text{gs}} = E_0 + \frac{1}{2}\sum_m \varepsilon_m$.

For numerical matrices, the quasiparticle energies are non-negative and sorted in ascending order.

Numerical fermionic matrices are diagonalized as Hermitian matrices; the eigenvectors x_m with positive eigenvalues define the quasiparticles, and the remaining columns are $Cx_m = \tau_x x_m^*$. Eigenvalues with $|\lambda| \le$ `"Tolerance"` $\times \max(1, \max|\lambda|)$ are zero modes; they are combined into fermions with $\varepsilon = 0$ in a basis invariant under C, so that T keeps the BdG structure. At $T = 0$ these modes are half occupied in `gaussianaverage`.

Numerical bosonic matrices are diagonalized by Colpa's method: Cholesky decomposition $M = K^\dagger K$ and diagonalization of $K\eta K^\dagger$. This requires M to be positive definite. Otherwise (dynamical instability, or zero modes such as Goldstone modes) the message `bogoliubov::posdef` is given and `$Failed` is returned.

Symbolic matrices are supported if the graph of nonzero matrix elements decomposes into blocks of at most two of the $2n$ components of Ψ. This covers singlet pairing ($c_\uparrow$ with $c_\downarrow^\dagger$), two-mode squeezing (a with $b^\dagger$), single-mode squeezing (b with $b^\dagger$), and the hybridization of two levels. Larger blocks give the message `bogoliubov::symbolic`. The symbolic results assume real diagonal elements and real quasiparticle energies; the energies are listed in the order of the modes and are not sorted, and their sign is not checked.

M must have the particle-hole symmetry of a BdG matrix, $\tau_x M^* \tau_x = -M$ (fermions) or M (bosons), where τ_x exchanges the two halves of Ψ; otherwise `bogoliubov::phs` is given.

`bogoliubov[H, modes]` computes `quadraticform[H, modes]` and diagonalizes the matrix; the modes are specified as in `quadraticform`. The result is `{ε, T, Egs}`. With `"Momentum" -> True`, M is a Bloch BdG matrix $H(k)$ for $\Psi_k = (c_k, c_{-k}^\dagger)$, whose particle-hole symmetry relates k and $-k$. All $2n$ columns of T are then eigenvectors of $H(k)$ (fermions, T unitary) or of $\eta H(k)$ (bosons, T paraunitary), and `ε` lists all $2n$ diagonal elements of $T^\dagger H(k) T$: the energies of the quasiparticles γ_k followed by $-\varepsilon(-k)$ (fermions) or $\varepsilon(-k)$ (bosons). For numerical fermionic matrices, the first n entries are the largest eigenvalues, with zero modes assigned to the particle columns first; if there are more than n positive or negative eigenvalues, `bogoliubov::count` is given.

The following options can be given:

`"Tolerance" 10^-10` relative tolerance for zero modes (fermions) and for the positive definiteness of M (bosons)

`"Momentum" False` whether M is a Bloch BdG matrix $H(k)$

Inexact or exact numerical matrices are diagonalized numerically; exact input is converted with `N`.

Example. On-site singlet pairing, $H = \xi(n_\uparrow + n_\downarrow) - \Delta(c_\uparrow^\dagger c_\downarrow^\dagger + c_\downarrow c_\uparrow)$. The quasiparticle energy is $E = \sqrt{\xi^2 + \Delta^2}$ and the ground-state energy $\xi - E$:

```
snegfermionoperators[c];
```

```
snegrealconstants[xi, Dl];
HS = xi number[c[1]] - Dl (nc[c[CR, 1, UP], c[CR, 1, DO]] + nc[c[AN, 1,
    DO], c[AN, 1, UP]]);
{ε, T, Egs} = bogoliubov[HS, {c[1]}];
{ε, Egs}
```

bogoliubovrules

`bogoliubovrules[T, {a1, ..., an}, g]` returns replacement rules that express the operators $a_i, a_i^\dagger$ in terms of the quasiparticle operators $g_m, g_m^\dagger$ for the Bogoliubov transformation matrix T.
`bogoliubovrules[T, {a1, ..., an}, {g1, g2, ...}]` uses the quasiparticle modes given as templates.
T is the $2n \times 2n$ matrix of the transformation $\Psi = T\Phi$, $\Psi = (a_1, \ldots, a_n, a_1^\dagger, \ldots, a_n^\dagger)^T$, $\Phi = (\gamma_1, \ldots, \gamma_n, \gamma_1^\dagger, \ldots, \gamma_n^\dagger)^T$, as returned by `bogoliubov`. The rules are

$$a_i \to \sum_{m=1}^{n} (T_{im}\gamma_m + T_{i,n+m}\gamma_m^\dagger), \qquad a_i^\dagger \to \sum_{m=1}^{n} (T_{n+i,m}\gamma_m + T_{n+i,n+m}\gamma_m^\dagger).$$

The modes a_i are specified as in `quadraticform` (`c` or `c[j...]`; for fermions with spin, the spin index is added as the last index and varies fastest). They must be single operators; modes given as functions (e.g. bonding orbitals) are rejected with `bogoliubovrules::mode`.
The quasiparticle operators γ_m are specified as
a boson symbol `g`: γ_m = `g[AN, m]`;
a spinless fermion symbol `g` (declared with `snegspinlessfermionoperators`): γ_m = `g[AN, m, 0]`;
a list of mode templates, as for the modes a_i, e.g. `{gam[1]}` for a spin-1/2 fermion gives $\gamma_1 = \gamma_{1\uparrow}, \gamma_2 = \gamma_{1\downarrow}$.
The number of quasiparticle modes must equal n; otherwise, or for a spinful fermion symbol, the message `bogoliubovrules::qp` is given. A matrix of the wrong dimension gives `quadraticoperator::dim`. In these cases `$Failed` is returned.
The quasiparticle operators must have the same statistics as the modes a_i; this is not checked.
The rules apply to expressions in which the operators appear explicitly, e.g. after `Expand`. Normal ordering of the result with respect to the quasiparticles is performed by `nc`, so `vev` of the result is the expectation value in the quasiparticle vacuum, i.e. the ground state for positive quasiparticle energies.

Example. On-site singlet pairing at $\xi = 4/5$, $\Delta = 3/5$; the quasiparticle energy is $E = 1$. In terms of the quasiparticles, $H = \gamma_1^\dagger\gamma_1 + \gamma_2^\dagger\gamma_2 + \xi - E$:

```
snegfermionoperators[c];
```

```
snegspinlessfermionoperators[g];
snegrealconstants[xi, Dl];
HS = xi number[c[1]] - Dl (nc[c[CR, 1, UP], c[CR, 1, DO]] + nc[c[AN, 1,
    DO], c[AN, 1, UP]]);
{ε, T, Egs} = bogoliubov[HS, {c[1]}];
rules = bogoliubovrules[T, {c[1]}, g];
Simplify[Expand[HS /. rules] /. {xi -> 4/5, Dl -> 3/5}]
```

gaussianaverage

`gaussianaverage[expr, M, {a1, ..., an}, T]` gives the expectation value of the operator expression *expr* in the thermal state at temperature T of the quadratic Hamiltonian with the Bogoliubov-de Gennes matrix M of the modes a_i.
`gaussianaverage[expr, H, {a1, ..., an}, T]` uses the quadratic Hamiltonian H itself.
`gaussianaverage[expr, Hk, {ck[a], ...}, T, "Momentum" -> k]` uses the Bloch BdG matrix $H(k)$ (or a momentum-space Hamiltonian) of a translation-invariant system; *expr* contains operators with momenta k and $-k$.
`gaussianaverage[{expr1, expr2, ...}, ...]` gives a list of expectation values.
The state is the Gibbs state $\rho = e^{-H/T}/\mathrm{Tr}\, e^{-H/T}$ of the quadratic Hamiltonian ($k_B = 1$). For `T = 0` it is the ground state. The chemical potential, if any, must be included in H; the average is grand canonical.
The modes are specified as in `quadraticform`: `c` or `c[j...]` (for fermions with spin, the spin index is added as the last index and varies fastest), or `Function[t, expr]`. The matrix M refers to the Nambu spinor $\Psi = (a_1, \ldots, a_n, a_1^\dagger, \ldots, a_n^\dagger)^T$; if H is given instead, M is computed with `quadraticform`.
The averages are computed from the correlation matrix $G_{pq} = \langle \Psi_p \Psi_q^\dagger \rangle$ with Wick's theorem. *expr* may be any polynomial in the mode operators, with products of any number of operators; terms with an odd number of operators give 0, and c-numbers are kept.
For numerical fermionic matrices, $G = (1 + e^{-M/T})^{-1}$ is computed directly from the eigenvectors of M; at $T = 0$, eigenvalues with $|\lambda| \le$ `"Tolerance"` (zero modes) are half occupied, which is the average over the degenerate ground states.
Otherwise $G = T_B D T_B^\dagger$ with the transformation T_B of `bogoliubov` and the quasiparticle occupations: Fermi function $1/(e^{\varepsilon/T} + 1)$ or Bose function $1/(e^{\varepsilon/T} - 1)$. Symbolic matrices are therefore restricted to blocks of at most two components, and bosonic matrices must be positive definite. At $T = 0$, the symbolic quasiparticle energies are assumed to be positive.
With `"Momentum" -> k`, the modes are orbital templates as in `bdghamiltonian`, and the averages are taken in the thermal state of $\frac{1}{2}\Psi_k^\dagger H(k) \Psi_k$ with

$\Psi_k = (c_{ka}, \ldots, c^\dagger_{-ka}, \ldots)$. The operators in *expr* must be components of Ψ_k or $\Psi^\dagger_k$, i.e. c_{ka}, c_{-ka} and their conjugates. For fermions, the occupations of the $2n$ columns are $1 - f(\varepsilon)$ for the particle and hole branches at $T > 0$, which accounts for the energies $-\varepsilon(-k)$ of the hole branch.

The following options can be given:

`"Momentum" None` momentum index k for Bloch BdG matrices

`"IndexPosition" 1` position of the momentum among the indexes after `CR/AN`, as in `bdghamiltonian`

`"Tolerance" 10^-10` tolerance for zero modes and for the diagonalization

Messages: `gaussianaverage::op` if *expr* contains operators that are not among the modes or terms that are not products of operators, `gaussianaverage::sum` for symbolic sums, `gaussianaverage::index` if the momentum is not a declared momentum index, and `quadraticoperator::dim` if the dimension of M does not match the modes. Failures of `quadraticform`, `bdghamiltonian`, and `bogoliubov` are passed on. In all these cases the result is `$Failed`.

Example. Ground-state averages for a tight-binding dimer with one electron per spin: the occupation of site 1 and the bond order $\langle c^\dagger_{1\uparrow} c_{2\uparrow}\rangle = 1/2$ of the bonding orbital:

```
snegfermionoperators[c];
H = -hop[c[1], c[2]];
gaussianaverage[{number[c[1]], nc[c[CR, 1, UP], c[AN, 2, UP]]}, H, {c
    [1], c[2]}, 0]
```

meanfield

`meanfield[H]` decouples all products of four or more fermion or boson operators in H into the Hartree-Fock-Bogoliubov mean-field Hamiltonian, with the averages of operator pairs written as `avg[...]`.

`meanfield[H, channels]` keeps only the contractions of the given channels: `"Hartree"`, `"Fock"`, `"Pairing"`, `"Normal"`, `"All"`, a list of these, or a selection function.

Every product of $2m \geq 4$ ladder operators $o_1 o_2 \cdots o_{2m}$ (as it appears in the normal-ordered H) is replaced by

$$\sum_{i<j} s_{ij}\, W(\mathrm{rest}_{ij})\, o_i o_j - (m-1)\, W(o_1 \cdots o_{2m}),$$

where W denotes the full Wick expansion with the allowed contractions $\langle o_a o_b\rangle$, rest_{ij} is the product without o_i and o_j, and s_{ij} is the fermionic sign of moving $o_i o_j$ to the front. The contractions are written as `avg[nc[oa, ob]]`.

For four operators this is the familiar decoupling $AB \to \langle A\rangle B + A\langle B\rangle - \langle A\rangle\langle B\rangle$, summed over the three pairings of the operators. For example, with all channels,

$n_\uparrow n_\downarrow \to \langle n_\downarrow\rangle n_\uparrow + \langle n_\uparrow\rangle n_\downarrow - \langle n_\uparrow\rangle\langle n_\downarrow\rangle - \langle c_\uparrow^\dagger c_\downarrow\rangle c_\downarrow^\dagger c_\uparrow - \ldots + \langle c_\downarrow c_\uparrow\rangle c_\uparrow^\dagger c_\downarrow^\dagger + \ldots$

The decoupling is exact for the energy: the average of the mean-field Hamiltonian in any Gaussian (quasi-free) state, with the averages `avg[...]` evaluated in the same state, equals the Wick average of H. The stationarity of this energy gives the Hartree-Fock-Bogoliubov equations; `meanfieldsolve` solves them self-consistently.

The channels select the allowed contractions $\langle o_1 o_2\rangle$ (o_1 to the left of o_2 in the product):

`"Hartree"` $\langle a^\dagger a\rangle$ of the same mode (equal operator labels)
`"Fock"` $\langle a^\dagger b\rangle$ of different modes, including spin-flip averages $\langle c_\uparrow^\dagger c_\downarrow\rangle$
`"Pairing"` anomalous averages $\langle ab\rangle$, $\langle a^\dagger b^\dagger\rangle$
`"Normal"` `"Hartree"` and `"Fock"`
`"All"` all contractions (default)
`{ch1, ch2, ...}` the union of the channels
`f` contractions for which `f[o1, o2]` gives `True`

Only operators of the same statistics are contracted (fermions with fermions, bosons with bosons). Terms with zero or two operators are kept unchanged.

The channel names refer to the operator labels, not to the physical process. In momentum space, the direct (Hartree) term of a density-density interaction appears as $\langle c_{k+q}^\dagger c_k\rangle$ and belongs to `"Fock"`; use `"Normal"` there and impose momentum conservation on the averages afterwards.

Symbolic sums are decoupled term by term inside the sum.

Terms with an odd number of operators (`meanfield::odd`), terms containing other operators such as spins or X-operators (`meanfield::op`), and terms that are not products of operators (`meanfield::term`) are kept unchanged with a message. An unknown channel gives `meanfield::channel` and `$Failed`.

The averages `avg[...]` are numbers for `nc`, `quadraticform`, and `bdghamiltonian`, so the mean-field Hamiltonian can be diagonalized directly. `avg` acts on the canonically ordered expression, so a contraction may appear as `1 - avg[...]` or with a sign. The result is expanded.

Example. Hartree decoupling of the Hubbard interaction $Un_\uparrow n_\downarrow$:

```
snegfermionoperators[c];
snegrealconstants[U];
meanfield[U hubbard[c[1]], "Hartree"]
```

meanfieldsolve

`meanfieldsolve[Hmf, {a1, a2, ...}, {avg[x1] -> v1, ...}]` solves the self-consistency equations for the averages `avg[...]` in the quadratic mean-field Hamiltonian *Hmf* of the modes a_i, starting from the given initial values, and returns an `Association` with the solution.

`meanfieldsolve[Hmf, modes]` starts from zero for all averages.
Hmf is a quadratic Hamiltonian whose coefficients depend on averages `avg[x]`, typically the result of `meanfield`. All other coefficients must be numerical; remaining symbols give the message `meanfieldsolve::numeric` and `$Failed`. The modes are specified as in `quadraticform`.
The iteration is: the averages are inserted in *Hmf*; its BdG matrix M (see `quadraticform`) gives the correlation matrix $G = \langle \Psi \Psi^\dagger \rangle$ of the thermal (or ground) state; the averages are recomputed from G with Wick's theorem; and the new values are mixed linearly with the old ones, $x \to x + \alpha(x_{\text{new}} - x)$. The iteration stops when $\max |x_{\text{new}} - x|$ is below `"Tolerance"`, or after `"MaxIterations"` steps with the message `meanfieldsolve::conv`.
Initial values: averages that are not given start from 0. A value can be given for an average or for its conjugate; the conjugate average gets the complex conjugate value unless it is given as well. Rules of the form `c avg[x] -> v`, as they arise from the canonical ordering of `avg` (e.g. `-avg[...]`), are converted to `avg[x] -> v/c`. Initial values for averages that do not occur in *Hmf* give the warning `meanfieldsolve::init`.
The result is an `Association` with the keys:
`"Averages"` rules `avg[x] -> value` for all averages in *Hmf*
`"Energy"` $\langle H_{\text{MF}} \rangle$ in the self-consistent state, which equals the Hartree-Fock(-Bogoliubov) energy $\langle H \rangle$ for `Hmf = meanfield[H]`; with a fixed particle number, the chemical-potential term is not included
`"QuasiparticleEnergies"` quasiparticle energies of $H_{\text{MF}} - \mu N$ from `bogoliubov` (`$Failed` if not converged)
`"ChemicalPotential"` μ for a fixed particle number, otherwise `None`
`"Iterations"` number of iterations
`"Converged"` `True` if the tolerance was reached
The following options can be given:
`"Temperature"` `0` temperature T of the thermal state ($k_B = 1$); 0 for the ground state
`"ParticleNumber"` `None` fixed fermion number N, adjusted with the chemical potential
`"Mixing"` `1/2` mixing parameter α
`"Tolerance"` `10^-10` convergence threshold for the averages
`"MaxIterations"` `1000` maximal number of iterations
Without `"ParticleNumber"`, the state is grand canonical with the chemical potential contained in *Hmf*.
With `"ParticleNumber" -> N` (fermions only, $0 \le N \le n$), the term $-\mu \hat{N}$ is added in every iteration. For number-conserving Hamiltonians at $T = 0$, the N lowest single-particle levels are filled and μ is set halfway between the highest occupied and the lowest unoccupied level; if the level at the Fermi energy is degenerate and

partially filled, its states are occupied equally (an ensemble average) with the message `meanfieldsolve::open`. Otherwise μ is determined by bisection so that $\langle \hat{N} \rangle = N$. For bosons, `meanfieldsolve::boson` is given; an invalid N gives `meanfieldsolve::number`.
At $T = 0$, fermionic zero modes are half occupied, as in `gaussianaverage`.
The self-consistent solution is a stationary point of the mean-field energy, not necessarily the minimum. Different initial values can converge to different solutions (e.g. paramagnetic and broken-symmetry); compare their energies. In particular, starting from zero averages preserves the symmetries of the Hamiltonian.
The solver works with finite sets of modes. Momentum-space problems can be treated on finite clusters (`sumCluster`), or through the self-consistency conditions obtained with `bdghamiltonian` and `gaussianaverage`.

Example. Hartree-Fock for the half-filled Hubbard dimer, $H = -t \sum_\sigma (c_{1\sigma}^\dagger c_{2\sigma} + \text{h.c.}) + U \sum_i n_{i\uparrow} n_{i\downarrow}$, at $U = 4t$, starting from an antiferromagnetic configuration:

```
snegfermionoperators[c];
H = -hop[c[1], c[2]] + 4 (hubbard[c[1]] + hubbard[c[2]]);
sol = meanfieldsolve[meanfield[H], {c[1], c[2]}, {avg[number[c[1], UP]]
    -> 0.9, avg[number[c[2], DO]] -> 0.9}, "ParticleNumber" -> 2];
Keys[sol]
```

quadraticform

`quadraticform[H, {a1, a2, ...}]` returns `{M, E0}`, the Bogoliubov-de Gennes (BdG) matrix M and the constant E_0 with $H = \frac{1}{2}\Psi^\dagger M \Psi + E_0$, $\Psi = (a_1, \ldots, a_n, a_1^\dagger, \ldots, a_n^\dagger)^T$, for a Hamiltonian H quadratic in the modes a_i.
`quadraticform[H, modes, "BdG" -> False]` returns `{h, C}` with $H = \sum_{ij} a_i^\dagger h_{ij} a_j + C$ for a number-conserving H.
`quadraticform[H, modes, "Linear" -> True]` returns `{M, f, E0}`, including the coefficients f_i of linear boson terms $\sum_i (f_i b_i^\dagger + f_i^* b_i)$.
A general quadratic Hamiltonian of n fermion or boson modes,

$$H = \sum_{ij} h_{ij} a_i^\dagger a_j + \frac{1}{2} \sum_{ij} (\Delta_{ij} a_i^\dagger a_j^\dagger + \Delta_{ij}^* a_j a_i) + C,$$

is written in the BdG (Nambu) form $H = \frac{1}{2}\Psi^\dagger M \Psi + E_0$. The Nambu spinor Ψ lists all annihilation operators first, followed by the creation operators in the same order. For fermions, the $2n \times 2n$ matrix M has the blocks h (upper left), Δ (upper right), $\Delta^\dagger$ (lower left), and $-h^T$ (lower right), with $h = h^\dagger$ and $\Delta^T = -\Delta$. The constant is $E_0 = C + \frac{1}{2}\mathrm{Tr}\, h$.

For bosons, the blocks are A, B, B^*, A^*, with $A = A^\dagger$ and $B^T = B$, and $E_0 = C - \frac{1}{2}\mathrm{Tr}\, A$.

The matrix elements are computed as $M_{pq} = \eta_p \eta_q [\Psi_p, [H, \Psi_q^\dagger]]_\mp$, with the anticommutator for fermions and the commutator for bosons; $\eta_p = 1$ for fermions, and $\eta = (1, \dots, 1, -1, \dots, -1)$ for bosons. M is therefore always in the symmetrized form above, independently of how the terms of H are ordered or split, e.g. $\Delta c_1^\dagger c_2^\dagger$ and $-\Delta c_2^\dagger c_1^\dagger$ give the same M.

The modes are specified in one of the following forms:

`c` or `c[j...]` a fermion level or a boson; for fermions with spin, the spin index is added as the last index and varies fastest, e.g. `{c[1], c[2]}` gives $(c_{1\uparrow}, c_{1\downarrow}, c_{2\uparrow}, c_{2\downarrow})$; spinless fermions get the index `0`

`Function[t, expr]` a mode given by its operator for `t = CR` and `t = AN`, e.g. a bonding orbital $(c_1 + c_2)/\sqrt{2}$

All modes must be of the same statistics (all fermions or all bosons).

The following options can be given:

`"BdG" True` if `False`, return the hopping matrix h $(n \times n)$ and the constant C instead of M and E_0

`"Linear" False` if `True`, return the linear boson coefficients f_i as the middle element

`"Check" True` check that M is Hermitian (and $g_i = f_i^*$ for the linear terms)

With `"BdG" -> False`, anomalous terms ($\Delta \neq 0$) are not allowed. For fermions $C = E_0 - \frac{1}{2}\mathrm{Tr}\, h$, for bosons $C = E_0 + \frac{1}{2}\mathrm{Tr}\, A$.

For fermions, `"Linear" -> True` returns a list of zeros as f; terms linear in the fermion operators are always rejected.

`quadraticform` fails with a message and returns `$Failed` if H contains terms of higher order (`quadraticform::order`), operators that are not among the modes (`quadraticform::rest`), linear fermion terms (`quadraticform::odd`), linear boson terms without `"Linear" -> True` (`quadraticform::linear`), anomalous terms with `"BdG" -> False` (`quadraticform::anom`), or symbolic sums (`quadraticform::sum`). A non-Hermitian M gives the warning `quadraticform::herm`, and the result is still returned.

Symbolic sums are not supported. For translation-invariant Hamiltonians in momentum space use `bdghamiltonian` or `blochhamiltonian`; finite clusters can be written explicitly with `sumCluster`.

Coefficients must be numbers in the sense of `isnumericQ`: declare parameters with `snegrealconstants` or `snegcomplexconstants`. Mean-field averages `avg[...]` are treated as numbers, so mean-field Hamiltonians from `meanfield` can be passed directly.

Inexact coefficients are allowed; rounding errors of order 10^{-10} are removed with `Chop`.

Example. The Kitaev chain with three sites, $H = \sum_{j=1}^{2}[-t(f_j^\dagger f_{j+1} + \text{h.c.}) + \Delta(f_j^\dagger f_{j+1}^\dagger + f_{j+1}f_j)] - \mu\sum_{j=1}^{3} n_j$:

```
snegspinlessfermionoperators[f];
snegrealconstants[t, mu, Dl];
H = Sum[-t hop[f[j], f[j + 1]] + Dl (nc[f[CR, j, 0], f[CR, j + 1, 0]] +
    nc[f[AN, j + 1, 0], f[AN, j, 0]]), {j, 2}] - mu Sum[number[f[j]], {j
    , 3}];
{M, E0} = quadraticform[H, {f[1], f[2], f[3]}];
MatrixForm[M]
```

quadraticoperator

`quadraticoperator[M, {a1, ..., an}, E0]` gives the operator $\frac{1}{2}\Psi^\dagger M\Psi + E_0$ for a $2n \times 2n$ Bogoliubov-de Gennes matrix M, with $\Psi = (a_1, \dots, a_n, a_1^\dagger, \dots, a_n^\dagger)^T$.
`quadraticoperator[h, {a1, ..., an}, E0]` gives $\sum_{ij} a_i^\dagger h_{ij} a_j + E_0$ for an $n \times n$ matrix h.
`quadraticoperator[M, modes]` is `quadraticoperator[M, modes, 0]`.
`quadraticoperator` is the inverse of `quadraticform`: with `{M, E0} = quadraticform[H, modes]`, `quadraticoperator[M, modes, E0]` reconstructs H; the same holds for the hopping form returned with `"BdG" -> False`.
The form is chosen by the dimension of the matrix: $2n \times 2n$ gives the BdG form, $n \times n$ the hopping form. Other dimensions give the message `quadraticoperator::dim` and `$Failed`.
The modes are specified as in `quadraticform`: `c` or `c[j...]` (for fermions with spin, the spin index is added as the last index and varies fastest; spinless fermions get the index `0`), or `Function[t, expr]` giving the operator for `t = CR, AN`.
In the BdG form, $\frac{1}{2}\Psi^\dagger M\Psi = \frac{1}{2}\sum_{pq} M_{pq}\Psi_p^\dagger\Psi_q$ is brought to normal order by `nc`. For a fermionic BdG matrix with the blocks h, Δ, $\Delta^\dagger$, $-h^T$ this gives $\sum_{ij} h_{ij}a_i^\dagger a_j + \frac{1}{2}\sum_{ij}(\Delta_{ij}a_i^\dagger a_j^\dagger + \text{h.c.}) - \frac{1}{2}\operatorname{Tr} h$; for a bosonic matrix with the blocks A, B, B^*, A^*, the constant is $+\frac{1}{2}\operatorname{Tr} A$.
The matrix is not checked for Hermiticity or for the BdG structure. A matrix without the BdG symmetry gives the operator $\frac{1}{2}\Psi^\dagger M\Psi$ as written; the symmetrized matrix is returned by `quadraticform`.
The result is expanded.

Example. A two-site hopping problem from its 2×2 matrix:

```
snegspinlessfermionoperators[f];
snegrealconstants[t, ε];
quadraticoperator[{{ε, -t}, {-t, ε}}, {f[1], f[2]}]
```

15.9 Effective Hamiltonians

eigenoperators

`eigenoperators[O, H0]` decomposes the operator O into eigenoperators of the commutator with *H0* and returns `{{omega1, O1}, {omega2, O2}, ...}` with $O = \sum_\omega O_\omega$ and $[H_0, O_\omega] = \omega O_\omega$.
O_ω changes the energy of H_0 by ω: if $|n\rangle$ is an eigenstate of H_0 with energy E_n, then $O_\omega|n\rangle$ is an eigenstate with energy $E_n + \omega$ (or zero). In the eigenbasis of H_0, O_ω contains the matrix elements $\langle m|O|n\rangle$ with $E_m - E_n = \omega$.
H_0 must be diagonal in the occupation numbers of the local modes: polynomials in fermion number operators (e.g. $\epsilon_d n_d + U n_{d\uparrow} n_{d\downarrow}$) or diagonal X-operators ($\sum_a E_a X^{aa}$). All other operators in O must be eigenoperators of the remaining part of H_0, $[H_0, o] = \lambda o$: band energies $\sum_k \epsilon_k n_k$, linear bosons $\omega b^\dagger b$, Zeeman terms BS^z.
For a normal-ordered monomial M and a polynomial $f(n)$ of the occupation numbers, $f(n)M = Mf(n + \Delta)$, where Δ is the change of the occupations by M. Hence $[H_0, M] = Mg(n)$ with a polynomial g in the occupations of the local modes not contained in M ("spectators"), and M is split with projectors onto the values of the spectator occupations. The components O_ω therefore contain products of number operators n and $1 - n$.
The energies ω are expressions in the parameters of H_0. The components are grouped by ω; components that vanish are dropped. The order of the list is not sorted.
The result is the basis of the Schrieffer-Wolff transformation in `schriefferwolff`: the first-order generator is $S_1 = \sum_{\omega \neq 0} V_\omega / \omega$ for the block-off-diagonal components of V.
Messages and failures (`$Failed`): `eigenoperators::diag` if H_0 is not diagonal or an operator is not an eigenoperator of H_0; `eigenoperators::sum` for symbolic sums (decompose the summand instead); `eigenoperators::term` for terms that are not products of operators; `eigenoperators::ordering` if local fermions are declared with an ordering other than the default `EMPTY`.
Coefficients may contain real functions such as `ε[k]` declared with `snegrealfunctions`.

Example. The hopping of a spin-up electron in the Hubbard dimer with $H_0 = U(n_{1\uparrow} n_{1\downarrow} + n_{2\uparrow} n_{2\downarrow})$ changes the energy by 0 or $\pm U$:

```
snegfermionoperators[c];
snegrealconstants[U];
H0 = U (hubbard[c[1]] + hubbard[c[2]]);
e = eigenoperators[nc[c[CR, 1, UP], c[AN, 2, UP]], H0];
e[[All, 1]]
```

schriefferwolff

`schriefferwolff[H0, V, P, n]` gives the contributions of orders 0, 1, ..., n in V to the effective Hamiltonian $PH_{\text{eff}}P$ obtained by the canonical Schrieffer-Wolff transformation of $H_0 + V$ with respect to the projector P.
The Hamiltonian $H = H_0 + V$ is block-diagonalized with respect to P and $Q = 1 - P$ by the unitary transformation $H_{\text{eff}} = e^S H e^{-S}$, $S = S_1 + S_2 + \cdots$, with S anti-Hermitian and block-off-diagonal ($PSP = QSQ = 0$), and S_n of order V^n. This canonical generator is unique.
The result is the list $\{PH_0P, PVP, PH^{(2)}_{\text{eff}}P, \ldots, PH^{(n)}_{\text{eff}}P\}$. At second order, $H^{(2)}_{\text{eff}} = \frac{1}{2}P[S_1, V]P$; for a degenerate P subspace with energy E_P, it equals $PVQ(E_P - H_0)^{-1}QVP$.
H_0 must be diagonal in the occupation numbers of the local modes, e.g. $\epsilon_d n_d + U n_{d\uparrow} n_{d\downarrow}$. All other operators in V must be eigenoperators of the rest of H_0: band energies $\sum_k \epsilon_k n_k$, linear bosons $\omega b^\dagger b$, Zeeman terms BS^z, and atomic Hamiltonians $\sum_a E_a X^{aa}$ are allowed.
P is written with number operators of the local modes or diagonal X-operators, e.g. `projector1[d[]]` for single occupancy of the level `d[]`, and must satisfy $P^2 = P$.
For translation-invariant Hamiltonians written as sums over a declared lattice index (see `sneglatticeindexes`), P can be given as a rule `{c, f}`, meaning the local projector `f[c[R]]` on every site. This is supported up to second order.
The results are operators that include the projector factors. `fermionstox` and `xtospin` convert them to X-operators and spin operators.
The following options can be given:
`"Generator" False` whether to also return the generators $S_1, \ldots, S_n$
`"Simplify" Simplify` function applied to the coefficients
With `"Generator" -> True`, the result is `{list, {S1, ..., Sn}}`.
If a component of V that connects P and Q has a vanishing energy denominator, the message `schriefferwolff::res` is issued and `$Failed` is returned. Symbolic denominators such as $\epsilon_k - \epsilon_d$ are only rejected if they vanish identically.
The run time grows quickly with the order and with the number of sites: the dimer to fourth order takes about a second, the 2×2 plaquette to fourth order (ring exchange) several minutes.

Example. Superexchange in the half-filled Hubbard dimer: the projector selects single occupancy of both sites.

```
snegfermionoperators[c];
snegrealconstants[t, U];
H0 = U (hubbard[c[1]] + hubbard[c[2]]);
V = -t hop[c[1], c[2]];
P = nc[projector1[c[1]], projector1[c[2]]];
```

```
heff = schriefferwolff[H0, V, P, 4];
```

schriefferwolffmatrix

`schriefferwolffmatrix[H0, V, pind, n]` performs the canonical Schrieffer-Wolff transformation of the matrix $H_0 + V$ to order n in V, with the low-energy subspace P spanned by the basis states with indexes *pind*, and returns the list $\{PH_0P, PVP, PH^{(2)}_{\text{eff}}P, \dots, PH^{(n)}_{\text{eff}}P\}$ of matrices.
The transformation is the same as in `schriefferwolff`: $H_{\text{eff}} = e^S(H_0 + V)e^{-S}$ with the anti-Hermitian, block-off-diagonal generator $S = S_1 + S_2 + \cdots$ ($PSP = QSQ = 0$, S_m of order V^m). Order by order, $[H_0, S_m] = PR_mQ + QR_mP$ and $H^{(m)}_{\text{eff}} = PR_mP$, where R_m is the part of order m of $e^{S'}He^{-S'}$, $S' = S_1 + \cdots + S_{m-1}$.
At second order, $H^{(2)}_{\text{eff}} = \frac{1}{2}P[S_1, V]P$, with the matrix elements

$$(H^{(2)}_{\text{eff}})_{ij} = \frac{1}{2}\sum_{q\in Q} V_{iq}V_{qj}\left(\frac{1}{E_i - E_q} + \frac{1}{E_j - E_q}\right), \qquad i,j \in P,$$

in the eigenbasis of H_0. For a degenerate P subspace with energy E_P this is $PVQ(E_P - H_0)^{-1}QVP$. The P subspace need not be degenerate; it only has to be separated from Q (no vanishing denominators $E_i - E_q$).
H0 and V are $d \times d$ matrices, and *pind* is a list of distinct indexes in $1, \dots, d$; Q is spanned by the other basis states. The results are $|P| \times |P|$ matrices in the basis given by *pind* (in that order).
H0 must not couple P and Q (`schriefferwolffmatrix::block`). If *H0* is diagonal, it may be symbolic. A non-diagonal *H0* must be numerical; it is diagonalized within the P and Q blocks, and the results are transformed back to the original basis (`schriefferwolffmatrix::diag` for symbolic non-diagonal matrices).
A nonzero element of a block-off-diagonal remainder with a vanishing energy denominator gives the message `schriefferwolff::res` and `$Failed`.
Invalid dimensions or indexes give `schriefferwolffmatrix::dim` and `$Failed`.
The following option can be given:
`"Generator" False` whether to also return the generators $S_1, \dots, S_n$ ($d \times d$ matrices)
With `"Generator" -> True`, the result is `{list, {S1, ..., Sn}}`.
Symbolic matrix elements are combined with `Together`. The cost grows with the order and the dimension; matrices with tens of states and orders up to about six are practical.
`schriefferwolffmatrix` is useful when H_0 is not diagonal in the occupation numbers, as required by `schriefferwolff`, for numerical studies, and as an independent check of operator results.

Example. A two-level system with the splitting Δ and the coupling v. The effective energy of the lower level reproduces the expansion of $(\Delta - \sqrt{\Delta^2 + 4v^2})/2$:

```
snegrealconstants[D1, v];
schriefferwolffmatrix[{{0, 0}, {0, D1}}, {{0, v}, {v, 0}}, {1}, 6]
```

15.10 Equations of Motion and Green's Functions

eomclosure

`eomclosure[A, H]` returns `{ops, M, a}` such that `A == a.ops` and the commutator of `ops[[i]]` with H is `M[[i]].ops`.
`ops` is a list of the distinct normal-ordered operator monomials generated from A by commutation with H. For r monomials, M is an $r \times r$ matrix and a has length r. The commutator is $[O_i, H]$, not $[H, O_i]$.
Rows give the images of the basis operators. Consequently the resolvent in `greensfunction` is contracted with a from the left. The ordering of monomials is not a physical quantum number.
Both arguments must be free of symbolic `sum`; A must have definite fermion parity. `eomclosure::sum`, `eomclosure::parity` and `eomclosure::term` indicate an unsupported input.
`"MaxOperators" 100` stop with `eomclosure::open` and `$Failed` if a further distinct monomial is required; reaching the limit is not a physical decoupling approximation

Example. For a Hubbard atom the interaction generates one additional operator, the annihilator conditioned on occupation by the opposite spin:

```
snegfermionoperators[d];
snegrealconstants[e, U];
h = e number[d[]] + U hubbard[d[]];
{ops, m, a} = eomclosure[d[AN, UP], h]
```

greensfunction

`greensfunction[A, B, H, z]` gives the retarded Green's function of A and B for the Hamiltonian H at complex frequency z, by solving the closed equations of motion. Time evolution is $A(t) = e^{iHt}Ae^{-iHt}$ with $\hbar = 1$. The retarded kernel is $-i\theta(t)$ times the expectation value of the anticommutator for odd A, or the commutator for even A. The Fourier convention is e^{izt} with positive imaginary part of z. Give B explicitly: it is not automatically conjugated.
For `eomclosure[A,H]` equal to `{ops,M,a}`, the result is $a \cdot (zI - M)^{-1}N$ with N_i the equal-time bracket of `ops[[i]]` with B, averaged in the chosen state. No factorization is made.
For $B = A^+$ the spectral density is $-\mathrm{Im}\, G(\omega + i0^+)/\pi$. On a finite cluster it consists of discrete lines. Use `lehmann` for their frequencies and weights.

Use explicit levels, not symbolic `sum` expressions. A must have definite fermion parity. A failed closure returns `$Failed`, not a truncated Green's function.

`"Average" avg` function applied to each equal-time bracket; by default residual expectation values are symbolic

`"MaxOperators" 100` maximum number of distinct operator monomials in the closure

Example. The two poles of the Hubbard atom are separated by the repulsion U. Their weights sum to one per spin:

In[1]:=
```
snegfermionoperators[d];
snegrealconstants[e, U];
h = e number[d[]] + U hubbard[d[]];
g = greensfunction[d[AN, UP], d[CR, UP], h, z];
Apart[g /. avg[number[d[], DO]] -> w, z]
```

Out[5]=
$$\frac{1-w}{z-e}+\frac{w}{-e-U+z}$$

lehmann

`lehmann[A, B, H, modes, T]` returns the poles and residues `{{omega1,w1}, ...}` of the retarded Green's function as a list ordered by increasing frequency, from numerical diagonalization of a finite fermionic Hamiltonian.

The propagator is $\sum_j w_j/(z-\omega_j)$. For energy eigenstates m, n, the frequency is $E_n - E_m$. The contribution to the residue is $\langle m|A|n\rangle\langle n|B|m\rangle$ times $\rho_m + \rho_n$ for odd A and $\rho_m - \rho_n$ for even A. The fermion parity of A must be definite.

At positive T, ρ_m are normalized Boltzmann weights with $k_B = 1$; at zero temperature the ground states share equal weight. Include the chemical potential in H when needed. This trace includes all particle numbers, not only the half-filled sector.

Supply a numerical Hermitian H and all the fermion levels appearing in H, A or B. Mode specifications are as for `quadraticform`; a spinful level contributes both spin orbitals. The Hamiltonian itself need not be quadratic.

For $B = A^\dagger$ and odd A the residues are nonnegative. Cross correlators can have signed or complex residues. The list contains no artificial broadening. If no transition survives the tolerance the result is `{}`.

`"Tolerance" 10^-9` absolute threshold for ground-state degeneracy at zero temperature, grouping close frequencies, and discarding small individual and grouped weights

Example. At particle-hole symmetry the Hubbard atom has two equal spectral lines per spin:

```
snegfermionoperators[d];
h = hubbard[d[]] - number[d[]]/2;
lehmann[d[AN, UP], d[CR, UP], h, {d}, 1/3]
```

liouvillian

`liouvillian[A, H, n]` returns the list starting with A and ending with the nth nested commutator with H on the right.
With $\mathcal{L}A = [A,H]$, the list is $\{A, \mathcal{L}A, \ldots, \mathcal{L}^n A\}$ after algebraic simplification. The order n is a nonnegative integer. For order zero the result is a one-element list.
The sign is appropriate to $i\,dA/dt = [A,H]$ for a time-independent Schrödinger-picture operator and $\hbar = 1$. The Liouvillian is an ordinary commutator even for fermions. The parity-dependent bracket enters only when taking a Green's function or moment.
Unlike `eomclosure`, this operation can retain symbolic momentum sums. It computes only the requested orders and does not require an exactly closed hierarchy.

Example. A free annihilator has Liouvillian eigenvalue e and its adjoint has eigenvalue $-e$:

```
snegfermionoperators[d];
snegrealconstants[e];
h = e number[d[]];
{liouvillian[d[AN, UP], h, 2], liouvillian[d[CR, UP], h, 1]}
```

spectralmoments

`spectralmoments[A, B, H, n]` gives the first $n+1$ coefficients in the large-frequency expansion of the retarded Green's function of A, B for H, starting at order $1/z$.
The coefficient of z^{-k-1} is the mean equal-time bracket of $\mathcal{L}^k A$ with B, where $\mathcal{L}A = [A,H]$. Use the anticommutator for odd A, and the commutator for even A.
Mixed parity in A produces `spectralmoments::parity` and `$Failed`.
For a diagonal fermionic propagator these are the moments of the spectral density per spin, not the moments of just its occupied part. For an ordinary canonical annihilator and its adjoint the zeroth moment is one at any temperature.
n must be a nonnegative integer. Symbolic `sum` expressions are allowed; one need not close the equations of motion to determine any fixed number of moments.
`"Average" avg` function used for equal-time averages, for instance a thermal trace on a small cluster

Example. The variance of the two atomic addition/removal frequencies is $U^2 w(1-w)$ with w the occupation of the opposite spin:

```
snegfermionoperators[d];
snegrealconstants[e, U];
h = e number[d[]] + U hubbard[d[]];
mom = spectralmoments[d[AN, UP], d[CR, UP], h, 2] /. avg[number[d[], DO
    ]] -> w
```

thermalaverage

`thermalaverage[X, H, modes, T]` returns the thermal expectation of X in a finite fermion system by numerical diagonalization of H. At `T = 0` it returns the equal-weight ground-state average.
The average is $\mathrm{Tr}(Xe^{-H/T})/\mathrm{Tr}(e^{-H/T})$ at positive temperature, with $k_B = 1$. Supply a numerical Hermitian Hamiltonian, including $-\mu N$ if appropriate. All charge sectors are included.
The list of fermion modes is specified as for `quadraticform`. Both spin projections of a declared spin-1/2 level are included. Operators in X and H must refer only to the supplied modes. Unsupported modes can produce `lehmann::mode` and `$Failed`. Interactions are allowed. In contrast, `gaussianaverage` uses Wick's theorem for a quadratic Hamiltonian and can treat many more modes. An exact thermal trace still uses floating-point energies and eigenstates here.
`"Tolerance" 10^-10` positive absolute energy tolerance: at zero temperature, states with energy strictly below the minimum plus this tolerance are equally weighted

Example. The ground state of the particle-hole symmetric repulsive atom is a doublet. The ensemble has zero magnetization but a finite squared moment:

In[1]:=
```
snegfermionoperators[d];
h = hubbard[d[]] - number[d[]]/2;
thermalaverage[#, h, {d}, 0] & /@ {spinz[d[]], pow[spinz[d[]], 2],
    hubbard[d[]]}
```

Out[3]= $\{0, 0.25, 0\}$

15.11 Output and Conversion

asciitosneg

`asciitosneg[string]` parses a string in the compact ASCII operator notation and returns the corresponding operator product.
`asciitosneg[expr]` converts all strings in the expression *expr*.

In the string, `"c+(i,j,...)"` denotes the creation operator `c[CR, i, j, ...]` and `"c(i,j,...)"` the annihilation operator `c[AN, i, j, ...]`; *c* must be a declared operator. The indexes are separated by commas and converted with `ToExpression`, so the spin can be given as `UP`/`DO` or 1/0, and symbolic indexes are allowed. `"c+()"` and `"c()"` are operators without indexes.
`"|i,j>"` denotes `ket[i, j]` and `"<i,j|"` denotes `bra[i, j]`.
Juxtaposed factors are multiplied with `nc`, so the result is brought to the canonical order.
Coefficients and sums are not parsed inside the string; give them outside, e.g. `2 "c+(1,1)" + "c+(2,1)"`.
`snegtoascii` performs the inverse conversion.

Example.

```
snegfermionoperators[c];
asciitosneg["c+(1,UP)c(2,DO)"]
```

PrettyOutput

`PrettyOutput` is a global setting: if `True` (the default), operators and products are formatted in the conventional notation, e.g. $c^{\dagger}_{1\uparrow} \cdot c_{2\downarrow}$.
With `PrettyOutput = True`, SNEG defines output formats for fermion and boson operators (daggers, spin arrows), spin operators (S^z_j), X-operators (X^{ab}_j), Grassmann conjugates ($\bar{\zeta}$), `nc` products (infix with center dots), `KroneckerDelta` (δ_{ij}), `UnitStep` ($\theta(x)$), `VACUUM` ($|\varnothing\rangle$), `dd`, `avg`, and others. The formats affect only the display; `InputForm` and `FullForm` show the expressions as they are.
The formats of an operator family are defined when it is declared. Setting `PrettyOutput = False` after loading SNEG disables the formats of operators declared afterwards; the formats of `nc` and the other functions are defined when SNEG is loaded.
To disable all formats, set Sneg`PrettyOutput = False`‘ before loading the package.
For LaTeX output, use `snegtex`.

Example.

```
snegfermionoperators[c];
{PrettyOutput, nc[c[CR, 1, UP], c[AN, 2, DO]]}
```

snegold2newrules

`snegold2newrules[{o1, o2, ...}, {n1, n2, ...}, {r1, r2, ...}]` gives the rules that express the old levels *o1*, *o2*, ... in terms of the new levels *n1*, *n2*, ..., where the new levels are defined by the linear combinations *r1*, *r2*, ... of the old levels.

The levels are given without the creation/annihilation index and without the spin index, e.g. `c[1]` or `e[]`. The definitions $n_i = \sum_j M_{ij} o_j$ are given as linear combinations of the old levels.
The transformation matrix M must be unitary; then $o_j = \sum_i (M^{-1})_{ji} n_i$, and the result is the list of rules `o_j -> ...`.
If the numbers of old and new levels differ, the text "Mismatching lengths." is printed and `Null` is returned. If M is not unitary, the text "The transformation rule is not unitary." is printed and `Null` is returned. For symbolic matrices, the unitarity check is not decided and is skipped.
The rules act on the levels. To transform operator expressions, they must be extended to the operators `c[CR, i, σ]` and `c[AN, i, σ]`; for a real matrix M, the same coefficients apply to creation and annihilation operators (see the examples). For a complex M, the coefficients of the creation operators are complex conjugated.

Example. Bonding and antibonding orbitals of two sites, $e = (c_1 + c_2)/\sqrt{2}$, $o = (c_1 - c_2)/\sqrt{2}$:

```
snegfermionoperators[c, e, o];
rules = snegold2newrules[{c[1], c[2]}, {e[], o[]}, {(c[1] + c[2])/Sqrt
    [2], (c[1] - c[2])/Sqrt[2]}]
```

snegtex

`snegtex[expr]` gives a string with the LaTeX form of the operator expression *expr*. Fermion operators are written as $c^\dagger_{i\uparrow}$ (`c^\dagger_{i\uparrow}`) and $c_{i\downarrow}$: the indexes after `CR`/`AN` are separated by commas, and the spin index of spin-1/2 operators is appended as `\uparrow` or `\downarrow`. Other spin values are converted with `TeXForm`; spinless fermions have no spin index.
Boson operators: $a^\dagger_q$. Majorana operators: γ_i. Spin operators `S[j, t]`: S^x_j, S^y_j, S^z_j, S^+_j, S^-_j for $t = 1, \ldots, 5$. X-operators `X[j, a, b]`: X^{ab}_j.
Operator names of one letter are set as math italics; longer names are set with `\mathrm`, e.g. `\mathrm{ph}^\dagger`.
Products `nc[a, b, ...]` are written as juxtaposed factors separated by spaces; factors that are sums are enclosed in `\left(...\right)`. Symbolic sums `sum[a, {k, ...}]` are written as `\sum_{k,...}`.
The rest of the expression (coefficients, fractions, functions) is converted with `TeXForm`, in which the operator products are atomic. Symbols with names of several letters appear as `\text{...}`, and Greek letters as LaTeX commands such as `\Delta` or `\xi`.
The result is a `String`; use it in a LaTeX math environment.

Example. The Hubbard dimer:

```
snegfermionoperators[c];
snegrealconstants[t, U];
snegtex[-t hop[c[1], c[2]] + U (hubbard[c[1]] + hubbard[c[2]])]
```

snegtoascii

`snegtoascii[expr]` converts the operators, bras, kets, and operator products in *expr* to a compact ASCII notation.
An operator `c[CR, i, j, ...]` is written as `"c+(i,j,...)"` and `c[AN, i, j, ...]` as `"c(i,j,...)"`. The indexes are converted with `ToString`, so the spin indexes `UP` and `DO` appear as 1 and 0. An operator without indexes is written as `"c+()"` or `"c()"`. `ket[i, j]` is written as `"|i,j>"` and `bra[i, j]` as `"<i,j|"`. Products `nc[...]` of converted factors are concatenated into one string.
Sums and coefficients are not converted: the result is an expression that contains one string for every operator product.
Factors that cannot be converted, such as undeclared symbols, are kept, and the product remains an `nc` of strings and expressions.
`asciitosneg` performs the inverse conversion.

Example.

```
snegfermionoperators[c];
snegtoascii[nc[c[CR, 1, UP], c[AN, 2, DO]]]
```

15.12 Extending SNEG

$SnegVersion

`$SnegVersion` is a string with the version number of the loaded SNEG package.
The version consists of the major, minor, and patch numbers, e.g. `"2.2.0"`. It is also printed when SNEG is loaded.
When reporting results, record `$SnegVersion` together with `$Version`.

Example.

```
$SnegVersion
```

acmt

`acmt[op1, op2]` gives the anticommutator $\{op_1, op_2\}$ of two fermion or Majorana operators *op1* and *op2*, which `nc` uses when it exchanges them.
When `nc` exchanges two adjacent operators of the same fermion family that are not in canonical order (see `snegOrderedQ`), it uses $x_1 x_2 = -x_2 x_1 + \{x_1, x_2\}$, with the

anticommutator given by `acmt[x1, x2]`. `acmt` thus defines the algebra of the operators; `antikomutator`, `komutator`, `vevwick`, and all other functions follow from it.
The built-in values are the canonical anticommutation relations:
operators `acmt`
`c[t1, j1, ...]`, `c[t2, j2, ...]` with `t1` ≠ `t2` (one `CR`, one `AN`) $\prod_n \delta_{j_n j'_n}$
(`KroneckerDelta` of the corresponding indexes); 0 if the numbers of indexes differ
`c[t, ...]`, `c[t, ...]` (same type) 0
`c[CR]`, `c[AN]` (no further indexes) 1
operators of different fermion families 0
Majorana operators `g[j1, ...]`, `g[j2, ...]` $\prod_n \delta_{j_n j'_n}$
Majorana operators of different families 0
bare Majorana symbols `g`, `g` 1
The type indexes are compared literally: two operators with different first indexes count as one creation and one annihilation operator.
Since the Kronecker deltas are written for symbolic indexes, the relations hold for arbitrary index values, e.g. $\{c_{k\sigma}, c^\dagger_{k'\sigma'}\} = \delta_{kk'}\delta_{\sigma\sigma'}$.
For undeclared symbols `acmt` remains unevaluated.

Example.

```
snegfermionoperators[c];
{acmt[c[AN, 1, UP], c[CR, 1, UP]], acmt[c[AN, 1, UP], c[CR, 2, UP]],
    acmt[c[CR, 1, UP], c[CR, 1, UP]]}
```

bosonQ

`bosonQ[a]` gives `True` if *a* is a boson operator symbol or a boson operator `a[...]`.
`bosonQ` is `True` for symbols declared with `snegbosonoperators` and for their applications `a[t, j, ...]`.
For other expressions, `bosonQ` remains unevaluated; use `TrueQ[bosonQ[x]]` in conditions.

Example.

```
snegbosonoperators[a];
{bosonQ[a], bosonQ[a[CR, 1]], TrueQ[bosonQ[u]]}
```

cmt

`cmt[op1, op2]` gives the commutator $[op_1, op_2]$ of two boson or spin operators *op1* and *op2*, which `nc` uses when it exchanges them.
When `nc` exchanges two adjacent operators of the same boson or spin family that are not in canonical order (see `snegOrderedQ`), it uses $x_1 x_2 = x_2 x_1 + [x_1, x_2]$ with the

commutator `cmt[x1, x2]`. `cmt` thus defines the algebra of the operators; `komutator` and the other functions follow from it.
Built-in values for bosons are the canonical commutation relations $[a_\alpha, a^\dagger_\beta] = \delta_{\alpha\beta}$, $[a_\alpha, a_\beta] = [a^\dagger_\alpha, a^\dagger_\beta] = 0$, with a product of `KroneckerDelta` for the indexes (0 if the numbers of indexes differ).
Built-in values for spin operators `S[j..., t]` (component `t` $= 1, ..., 5$ for $x, y, z, +, -$) are the su(2) relations on the same site, $[S^a, S^b] = i\epsilon_{abc}S^c$, $[S^z, S^\pm] = \pm S^\pm$, $[S^+, S^-] = 2S^z$, and the mixed relations such as $[S^+, S^x] = S^z$. For different sites with symbolic indexes, the commutator is multiplied by $\delta_{jj'}$. Spin operators of different symbols commute.
Operators of different boson families commute; this is built into `nc` and does not use `cmt`.
For undeclared symbols `cmt` remains unevaluated.

Example.

```
snegbosonoperators[a];
{cmt[a[AN, 1], a[CR, 1]], cmt[a[CR, 1], a[AN, 1]], cmt[a[AN, 1], a[CR,
    2]], cmt[a[AN, 1], a[AN, 2]]}
```

fermionQ

`fermionQ[c]` gives `True` if c is a fermion operator symbol or a fermion operator `c[...]`.
`fermionQ` is `True` for symbols declared with `snegfermionoperators` or `snegspinlessfermionoperators` and for their applications `c[t, j, ...]`.
Majorana operators are not fermions in this sense (`majoranaQ`), nor are Grassmann numbers (`grassmanQ`).
For other expressions, `fermionQ` remains unevaluated; use `TrueQ[fermionQ[x]]` in conditions.
Many SNEG functions select their rules with patterns such as `op_?fermionQ[j___]`.
The spin of a fermion family is `spinof[c]`.

Example.

```
snegfermionoperators[c];
snegspinlessfermionoperators[f];
{fermionQ[c], fermionQ[c[CR, 1, UP]], fermionQ[f[AN, 1, 0]]}
```

grassmanQ

`grassmanQ[x]` gives `True` if x is a Grassmann number declared with `sneggrassmanconstants`, or the conjugate `conj[x]` of one.

`grassmanQ` is `True` for the declared symbols and their conjugates. For other expressions, including products of Grassmann numbers, it remains unevaluated; use `TrueQ[grassmanQ[x]]` in conditions.
Grassmann numbers are neither operators (`operatorQ`) nor scalars (`isnumericQ`).

Example.

```
sneggrassmanconstants[x];
{grassmanQ[x], grassmanQ[conj[x]], TrueQ[grassmanQ[y]], isnumericQ[x]}
```

isannihilation

`isannihilation[op]` gives `True` if the fermion or boson operator *op* annihilates the reference state of its family, i.e. if it is an annihilation operator in the sense of the normal order.
The reference state is set by `ordering`:
`ordering[c] isannihilation[c[t, k, ...]]`
EMPTY (default) `t == AN`
SEA `t == AN` and $k > 0$, or `t == CR` and $k < 0$
NONE `t == AN`
bosons `t == AN`
An operator with `isannihilation` `True` acting on `VACUUM` gives zero, and so does `vev` of a product ending with it.
See `iscreation` for the details and the use as an extension point.

Example.

```
snegfermionoperators[c];
{isannihilation[c[AN, 1, UP]], isannihilation[c[CR, 1, UP]]}
```

iscreation

`iscreation[op]` gives `True` if the fermion or boson operator *op* creates a (quasi)particle when acting on the reference state of its family, i.e. if it is a creation operator in the sense of the normal order.
The reference state is set by `ordering`:
`ordering[c] iscreation[c[t, k, ...]]`
EMPTY (default) `t == CR`
SEA `t == CR` and $k > 0$, or `t == AN` and $k < 0$
NONE `t == CR`
bosons `t == CR`
For SEA, k is the second index (the momentum or energy measured from the Fermi level). For a symbolic k without a declared sign, the result is a symbolic condition

such as `k > 0`; for $k = 0$ it is `False`, as is `isannihilation`. The role is then undetermined.
`iscreation` and `isannihilation` are used by `snegOrderedQ` (normal order), by `vev` and `vevwick` (an operator that creates a quasiparticle at the left end of a product gives zero in the expectation value), and by the rules for `VACUUM`.
For other operators, e.g. spin operators or undeclared symbols, `iscreation` remains unevaluated.

Example.

```
snegfermionoperators[c];
{iscreation[c[CR, 1, UP]], iscreation[c[AN, 1, UP]]}
```

isnumericQ

`isnumericQ[x]` gives `True` if x is a scalar (c-number) that `nc` and the other linear functions of SNEG factor out of operator products, and `False` otherwise.
`isnumericQ[x]` is `True` if every subexpression of x (every element of `Level[x, {0, Infinity}]`) is
a numeric quantity in the sense of `NumericQ`, e.g. `2`, `I`, `Pi`, `Sqrt[2]`;
a symbol declared with `snegrealconstants`, `snegcomplexconstants`, `snegintegerconstants`, or `snegfreeindexes`, or an application `f[...]` of a function declared with `snegrealfunctions` or `snegcomplexfunctions`;
the head of an expression with the attribute `NumericFunction`, such as `Plus`, `Times`, `Power`, `Sin`, `Exp`, `Conjugate`, `Abs`. SNEG gives this attribute also to `KroneckerDelta`, `UnitStep`, and `levicivita`.
Expectation values `avg[...]` count as scalars, irrespective of their argument.
Operators, Grassmann numbers (`sneggrassmanconstants`), undeclared symbols, and functions without the `NumericFunction` attribute, e.g. `DiracDelta`, are not scalars. Symbolic sums `sum[...]` are scalars only if their summand is a scalar.
`isnumericQ` is used by `nc`, `dd`, `vev`, `conj`, and all functions declared with `sneglinearoperator`, `sneglinearoperatorFirst`, and `snegsesquilinearoperator` to decide which factors are pulled out. A symbol that is not declared as a constant is treated as an operator of unknown type: it stays inside the products, which are then not brought to normal order.

Example.

```
snegrealconstants[t];
snegcomplexconstants[z];
snegfermionoperators[c];
{isnumericQ[t], isnumericQ[2 t + I z^2], isnumericQ[u], isnumericQ[c[CR,
     1, UP]]}
```

majoranaQ

majoranaQ[g] gives True if *g* is a Majorana operator symbol or a Majorana operator g[...].
majoranaQ is True for symbols declared with snegmajoranaoperators and for their applications g[j, ...].
For other expressions, majoranaQ remains unevaluated; use TrueQ[majoranaQ[x]] in conditions. Majorana operators are not fermionQ.

Example.

```
snegmajoranaoperators[g];
snegfermionoperators[c];
{majoranaQ[g], majoranaQ[g[1]], TrueQ[majoranaQ[c]]}
```

operatorQ

operatorQ[op] gives True if *op* is a declared operator symbol or an operator op[...] of a declared family.
operatorQ is True for symbols declared with snegfermionoperators, snegspinlessfermionoperators, snegbosonoperators, snegmajoranaoperators, snegspinoperators, and snegxoperators, and for their applications op[j, ...].
For other expressions, operatorQ remains unevaluated (it is not False), except for operatorQ[nc], which is False. Use TrueQ[operatorQ[x]] in conditions.
operatorQ tests a single operator; products, sums, and Grassmann numbers are not operators in this sense.
The declaration functions set operatorQ[op] as an upvalue of op; for a new operator type, set op /: operatorQ[op] = True.

Example.

```
snegfermionoperators[c];
snegbosonoperators[a];
{operatorQ[c], operatorQ[c[CR, 1, UP]], operatorQ[a[AN]]}
```

scalar

scalar[x] marks *x* as a scalar (c-number) coefficient, which is factored out of operator products without further tests.
nc always factors scalar[x] out of products. Products and sums of scalar wrappers are combined: scalar[a] scalar[b] is scalar[a b], scalar[a] + scalar[b] is scalar[a + b], and a factor *z* with isnumericQ[z] is absorbed, z scalar[b] is scalar[z b].

`scalar[0]` is 0. `conj[scalar[x]]` is `scalar[conj[x]]`, and `vev[scalar[x]]` is x. Symbolic sums are pulled out: `scalar[a sum[b, {k}]]` is `sum[scalar[a b], {k}]`. `scalar` is used to speed up calculations with long coefficients, in particular in symbolic sums, where the test `isnumericQ` of every factor is expensive, and by functions such as `scalarproductvc`.
The replacement rules `rulestripscalar` (`scalar[x] :> x`) and `rulesimplifyscalar` (`scalar[x] :> scalar[Simplify[x]]`) remove or simplify the wrappers.
`isnumericQ[scalar[x]]` is `False`; the wrapper is recognized by explicit rules of `nc`, `conj`, `vev`, and the sum functions.

Example.

```
snegfermionoperators[c];
snegrealconstants[t];
nc[c[AN, 1, UP], scalar[t] c[CR, 1, UP]]
```

sneglinearoperator

`sneglinearoperator[f]` makes the function f linear in each of its arguments. After `sneglinearoperator[f]`, the following rules hold for scalars z (see `isnumericQ`) and arbitrary expressions a, b:
$f(\dots, a+b, \dots) = f(\dots, a, \dots) + f(\dots, b, \dots)$;
$f(\dots, z\,a, \dots) = z\,f(\dots, a, \dots)$;
$f(\dots, z, \dots) = z\,f(\dots, \dots)$: an argument that is a scalar is factored out and removed.
The last rule corresponds to the convention for products, where a scalar factor is a product with the identity: `nc[a, 2, b]` is `2 nc[a, b]`. It makes `sneglinearoperator` suitable for product-like functions with a variable number of arguments, such as `nc` and `dd`, which are declared in this way.
The rules are added as definitions of f; further definitions for the values on operators can be given before or after the declaration. `sneglinearoperator` returns `{Null}`.

Example.

```
snegfermionoperators[c];
snegrealconstants[t];
snegcomplexconstants[z];
sneglinearoperator[tensor];
tensor[t c[CR, 1, UP] + 2 c[CR, 2, UP], z c[AN, 1, UP]]
```

sneglinearoperatorFirst

`sneglinearoperatorFirst[f]` makes the function f linear in its first argument, with $f(z) = z$ for scalars z.

After `sneglinearoperatorFirst[f]`, the following rules hold for scalars z (see `isnumericQ`), Grassmann numbers ξ, and arbitrary expressions a, b:
$f(a+b,\ldots) = f(a,\ldots) + f(b,\ldots)$;
$f(z\,a,\ldots) = z\,f(a,\ldots)$;
$f(z) = z$, and $f(z,b,\ldots) = z\,f(b,\ldots)$;
$f(\xi) = \xi$ and $f(\xi a) = \xi f(a)$ for a product `nc[ξ, a]` starting with a Grassmann number.
These are the rules of a normalized linear functional, such as an expectation value $\langle A\rangle$ with $\langle 1\rangle = 1$. In SNEG, `vev`, `vevwick`, `avg`, and `zeroonvac` are declared in this way.
The further arguments, if any, are parameters (e.g. a state or a temperature) and are not expanded.
The rules are added as definitions of f; the values on operator products are given by further definitions. `sneglinearoperatorFirst` returns `{Null}`.

Example.

```
snegfermionoperators[c];
snegrealconstants[t];
sneglinearoperatorFirst[ev];
ev[3 + t number[c[1]]]
```

snegOrderedQ

`snegOrderedQ[op1, op2]` gives `True` if the operators *op1* and *op2* are in the canonical order, i.e. if the product `nc[op1, op2]` is left as it is, and `False` if `nc` exchanges them. `nc` calls `snegOrderedQ` for every pair of adjacent operators. If two operators of the same family are not ordered, it exchanges them using the (anti)commutator: $x_1x_2 = -x_2x_1 + \{x_1,x_2\}$ for fermions, with the anticommutator `acmt[x1, x2]`, and $x_1x_2 = x_2x_1 + [x_1,x_2]$ for bosons and spin operators, with the commutator `cmt[x1, x2]`. For fermions of different families, which anticommute, it gives $x_1x_2 = -x_2x_1$. The repeated exchanges sort the whole product.
The built-in rules are:
operators `snegOrderedQ[x1, x2]`
fermions, same symbol, `ordering` EMPTY `OrderedQ[{x1, x2}]`: creation operators (`CR` $= 0$) before annihilation operators (`AN` $= 1$), then by the remaining indexes
fermions, same symbol, `ordering` SEA `True` if *x1* is a quasiparticle creation and *x2* an annihilation operator, `False` in the opposite case, otherwise `OrderedQ[{x1, x2}]` (see `ordering`)
fermions, `ordering` NONE (either operator) `True`
fermions, different symbols by the roles `iscreation` and `isannihilation` of each operator in its own family (creation first); equal or unknown roles are sorted by the

raw `CR/AN` index and `OrderedQ`
bosons, same symbol `OrderedQ[{x1, x2}]`, or `True` for `ordering[a] = NONE`
Majorana fermions, spin operators, same symbol `OrderedQ[{x1, x2}]`
For other pairs, e.g. undeclared symbols, `snegOrderedQ` remains unevaluated and the product is not reordered. Products of bosons of different symbols, of Majorana fermions of different symbols, and of operators of different kinds (bosons with fermions, spins with other operators, X-operators) are ordered by separate rules of `nc` that do not use `snegOrderedQ`.
`nc` exchanges two operators of the same family only if `snegOrderedQ` gives `False` (more precisely, if `!snegOrderedQ[x1, x2]` evaluates to `True`); a result that is neither `True` nor `False` leaves the product unchanged.

Example.

```
snegfermionoperators[c];
{snegOrderedQ[c[CR, 1, UP], c[AN, 1, UP]], snegOrderedQ[c[AN, 1, UP], c[
    CR, 1, UP]]}
```

snegsesquilinearoperator

`snegsesquilinearoperator[f]` makes the function f of two arguments antilinear in the first and linear in the second argument.
After `snegsesquilinearoperator[f]`, the following rules hold for scalars z (see `isnumericQ`):
$f(z\,a,b) = z^* f(a,b)$ and $f(a,z\,b) = z f(a,b)$;
$f(a_1+a_2,b) = f(a_1,b)+f(a_2,b)$ and $f(a,b_1+b_2) = f(a,b_1)+f(a,b_2)$;
$f(0,b) = f(a,0) = 0$.
This is the convention of Dirac's notation, $\langle a|b\rangle$, used in physics; in the mathematical literature, inner products are often linear in the first argument.
The complex conjugate is computed with `conj`, so that declared real constants are invariant.
In SNEG, `scalarproduct` and `scalarproductvc` are declared in this way.
The rules are added as definitions of f; the values on basis vectors are given by further definitions. `snegsesquilinearoperator` returns `{Null}`.

Example.

```
snegrealconstants[t];
snegcomplexconstants[z];
snegsesquilinearoperator[sp];
sp[z a + t b, I a]
```

spinQ

spinQ[S] gives True if S is a spin operator symbol or a spin operator S[...].
spinQ is True for symbols declared with snegspinoperators and for their applications S[j, ..., t].
For other expressions, spinQ remains unevaluated; use TrueQ[spinQ[x]] in conditions.
The spin operators of fermion levels (spinz[c[]], ...) are operator expressions, not spin operators in this sense.
The spin size is spinof[S].

Example.

```
snegspinoperators[{S, 1/2}];
{spinQ[S], spinQ[S[1, 3]], spinof[S]}
```

15.13 Additional interfaces

fixspinket

fixspinket[ket[s], s, sz] gives the spin ket ket[sz] of the multiplet with total spin s, obtained from the highest-weight ket ket[s].
fixspinket is the counterpart of fixspin for spin kets: it shifts the spin projection in single-slot kets ket[m] by $sz - s$. Kets with several slots are returned unchanged.
fixspinket[k, s, s] is k. For $sz > s$ the expression remains unevaluated.
fixspin and fixspinket are used as arguments of SU2basistensorproduct to combine a fermionic basis with an impurity spin into a basis with good total spin, as done in NRG Ljubljana.

Example.

```
{fixspinket[ket[1], 1, 0], fixspinket[ket[1], 1, -1], fixspinket[ket
    [1/2], 1/2, -1/2]}
```

listspincomponentindexes

listspincomponentindexes is the list of the symbols declared as spin component indexes with snegspincomponentindexes.
The list is empty when the package is loaded and is extended by each declaration; reloading the package keeps the existing list.
The spin component indexes are also summation indexes and appear in listfreeindexes.

Example.

```
snegspincomponentindexes[a, b];
listspincomponentindexes
```

rulesimplifyscalar

`expr /. rulesimplifyscalar` applies `Simplify` to the contents of all scalar markers `scalar[x]` in *expr*.
`rulesimplifyscalar` is `{scalar[x_] :> scalar[Simplify[x]]}`. The marker is kept; use `rulestripscalar` to remove it. See `rulestripscalar` for the meaning of `scalar`.

Example.

```
snegfermionoperators[c];
snegrealconstants[t];
e = scalar[Sin[t]^2 + Cos[t]^2] c[CR, 1, UP]
```

spinketbraM

`spinketbraM[s]` gives the lowering operator S^- of the spin-s operator in the bra-ket representation.
`spinketbraM[s, pos]` places the spin in the slots specified by the position array *pos*. The result is $\sum_{mm'} |m\rangle M_{mm'} \langle m'|$, written with `nc[ket[m], bra[m']]`, with the matrix `spinmatrixM[s]`. See `spinketbraX` for the position arrays and the use of the bra-ket representation.
s must be an integer or a half-integer.

Example.

```
spinketbraM[1/2]
```

spinketbraOPS

`spinketbraOPS[s]` gives the list $\{S^x, S^y, S^z, S^+, S^-\}$ of the spin-s operators in the bra-ket representation.
`spinketbraOPS[s, pos]` places the spin in the slots specified by the position array *pos*.
The order corresponds to the component indexes 1, ..., 5 of abstract spin operators `S[j, t]` (see `snegspinoperators`), so that `Thread[Table[S[t], {t, 5}] -> spinketbraOPS[s]]` maps a single spin to the bra-ket representation; `spinketbrarules` generalizes this to several spins.
See `spinketbraX` for the position arrays.

Example.

```
spinketbraOPS[1/2]
```

spinketbraP

`spinketbraP[s]` gives the raising operator S^+ of the spin-s operator in the bra-ket representation.
`spinketbraP[s, pos]` places the spin in the slots specified by the position array *pos*. The result is $\sum_{mm'} |m\rangle M_{mm'} \langle m'|$, written with `nc[ket[m], bra[m']]`, with the matrix `spinmatrixP[s]`. See `spinketbraX` for the position arrays and the use of the bra-ket representation.
s must be an integer or a half-integer.

Example.
spinketbraP[1/2]

spinketbraXYZ

`spinketbraXYZ[s]` gives the list $\{S^x, S^y, S^z\}$ of the spin-s operators in the bra-ket representation.
`spinketbraXYZ[s, pos]` places the spin in the slots specified by the position array *pos*.
`spinketbraXYZ[s, pos]` is `{spinketbraX[s, pos], spinketbraY[s, pos], spinketbraZ[s, pos]}`; see `spinketbraX`.

Example.
spinketbraXYZ[1/2]

spinketbraY

`spinketbraY[s]` gives the y component of the spin-s operator in the bra-ket representation.
`spinketbraY[s, pos]` places the spin in the slots specified by the position array *pos*. The result is $\sum_{mm'} |m\rangle M_{mm'} \langle m'|$, written with `nc[ket[m], bra[m']]`, with the matrix `spinmatrixY[s]`. See `spinketbraX` for the position arrays and the use of the bra-ket representation.
s must be an integer or a half-integer.

Example.
spinketbraY[1/2]

spinketbraZ

`spinketbraZ[s]` gives the z component of the spin-s operator in the bra-ket representation.
`spinketbraZ[s, pos]` places the spin in the slots specified by the position array *pos*.

The result is $\sum_{mm'} |m\rangle M_{mm'} \langle m'|$, written with `nc[ket[m], bra[m']]`, with the matrix `spinmatrixZ[s]`, $M_{mm'} = m\,\delta_{mm'}$. See `spinketbraX` for the position arrays and the use of the bra-ket representation.
s must be an integer or a half-integer.

Example.

```
spinketbraZ[1/2]
```

spinmatrixM

`spinmatrixM[s]` gives the $(2s+1) \times (2s+1)$ matrix of the spin-lowering operator S^- in the spin-s representation.
The basis is $|s, m\rangle$ with $m = s, s-1, \ldots, -s$; $\langle m-1|S^-|m\rangle = \sqrt{s(s+1) - m(m-1)}$, so the nonzero elements are on the first subdiagonal. See `spinmatrixX` for the conventions.
s must be an integer or a half-integer; otherwise the expression remains unevaluated.

Example.

```
spinmatrixM[1] // MatrixForm
```

spinmatrixP

`spinmatrixP[s]` gives the $(2s+1) \times (2s+1)$ matrix of the spin-raising operator S^+ in the spin-s representation.
The basis is $|s, m\rangle$ with $m = s, s-1, \ldots, -s$; $\langle m+1|S^+|m\rangle = \sqrt{s(s+1) - m(m+1)}$, so the nonzero elements are on the first superdiagonal. See `spinmatrixX` for the conventions.
s must be an integer or a half-integer; otherwise the expression remains unevaluated.

Example.

```
spinmatrixP[1] // MatrixForm
```

spinmatrixY

`spinmatrixY[s]` gives the $(2s+1) \times (2s+1)$ matrix of the spin component S^y in the spin-s representation.
The basis is $|s, m\rangle$ with $m = s, s-1, \ldots, -s$; $S^y = (S^+ - S^-)/(2i)$ is imaginary and antisymmetric. See `spinmatrixX` for the conventions.
s must be an integer or a half-integer; otherwise the expression remains unevaluated.

Example.

```
spinmatrixY[1] // MatrixForm
```

spinmatrixZ

`spinmatrixZ[s]` gives the $(2s+1) \times (2s+1)$ matrix of the spin component S^z in the spin-s representation.
The basis is $|s, m\rangle$ with $m = s, s-1, \ldots, -s$; $S^z = \mathrm{diag}(s, s-1, \ldots, -s)$. See `spinmatrixX` for the conventions.
s must be an integer or a half-integer; otherwise the expression remains unevaluated.

Example.

In[1]:=
```
spinmatrixZ[1] // MatrixForm
```

Out[1]=
$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

A Conventions at a glance

Levels and indices

Input	Meaning
`c[i]`	a spinful level, without operator-type or spin index
`c[CR,i,UP]`	creation of its up electron
`c[AN,i,DO]`	annihilation of its down electron
`number[c[i]]`	sum over spin of the occupations
`hubbard[c[i]]`	$n_{i\uparrow}n_{i\downarrow}$ without U
`spinplus[c[i]]`	$c^\dagger_{i\uparrow}c_{i\downarrow}$
`S[i,4]`, `S[i,5]`	raising and lowering of a declared spin

The internal type indices are `CR`=0 and `AN`=1; spin-1/2 labels are `UP`=1, `DO`=0. Do not infer bit order by sorting these integers: consult the current `BASIS`. A bare boson normally needs no spin index. For spinless fermions use the mode convention of the constructor in question; the generated explicit operator may carry a trailing zero.

Order, charge and conjugation

Normal ordering is a rule for reducing an operator, not an expectation value. `conj` reverses the order of factors and changes their type. Complex coefficients must be conjugated too. The empty state has $Q = -L$ for L spin-degenerate levels; half filling has $Q = 0$. Highest-weight spin bases represent each complete spin multiplet once. Reflection parity acts on states including the fermionic permutation sign. Fermion parity $(-1)^N$ and spatial-reflection parity need not agree. Energy gaps must specify which quantum numbers are held fixed.

Bloch, Nambu and retarded conventions

An annihilator in real space has Fourier phase $+ikR$ and normalization $N_s^{-1/2}$. Ordered orbitals within a cell fix the rows of $h(k)$. Including their positions changes the phases of off-diagonal elements, but neither the eigenvalues nor gauge-invariant observables. In the particle-first Nambu spinor the hole block uses $-h^T(-k)$ for fermions; the prefactor is $1/2$ when summing the full doubled basis.

For $\operatorname{Im} z > 0$ the transform e^{izt} of the retarded kernel gives poles at $E_n - E_m$, with denominator $z - (E_n - E_m)$. Fermionic diagonal spectral density integrates to one per canonical mode including both signs of frequency. Use $\mathcal{L}A = [A, H]$; exchanging the order of the commutator reflects all pole frequencies.

B Extending the operator algebra

B.1 Normal order as a reduction rule

To add an algebra, specify which pairs are out of order and what replaces such a pair. For odd a, b the replacement is $ab = -ba + \{a, b\}$. For bosons use $ab = ba + [a, b]$. A scalar overlap on the right lowers the operator degree and can therefore be factored out of `nc`. A new commutator rule without an ordering that eventually stops interchanging pairs can instead give infinite recursion. In particular do not define both directions of a swap to call each other.

`ordering` sets the reference convention per family; `snegOrderedQ` is the predicate for canonical order. `acmt` and `cmt` supply the elementary brackets, whereas `antikomutator` and `komutator` construct brackets of arbitrary products from the algebra. Test a rule in the declared operator family first, preferably as an upvalue of its head; a broad downvalue on a public symbol can change every subsequent calculation in that kernel.

Case study: A continuum field with a distributional overlap

Field operators of a continuum, $\{\psi(x), \psi^\dagger(y)\} = \delta(x-y)$. The anticommutator is redefined as an upvalue of `psi`, and `DiracDelta` is declared as a scalar:

In[1]:=
```
snegspinlessfermionoperators[psi];
snegrealconstants[x, y, z];
snegnonopQ[DiracDelta] = True;
psi /: acmt[psi[AN, x_], psi[CR, y_]] := DiracDelta[x - y];
psi /: acmt[psi[CR, x_], psi[AN, y_]] := DiracDelta[x - y];
nc[psi[AN, x], psi[CR, y], psi[CR, z]]
```

Out[6]= $\mathrm{psi}^\dagger_y \mathrm{psi}^\dagger_z \mathrm{psi}_x - \mathrm{psi}^\dagger_y \delta(x-z) + \mathrm{psi}^\dagger_z \delta(x-y)$

The two-particle vacuum amplitude $\langle 0|\psi(x)\psi(z)\psi^\dagger(z)\psi^\dagger(y)|0\rangle$:

```
In[7]:= vevwick[nc[psi[AN, x], psi[AN, z], psi[CR, z], psi[CR, y]]]
```

Out[7]= $\delta(0)\delta(x-y) - \delta(x-z)\delta(z-y)$

B.2 Scalars and linear maps

The predicate `isnumericQ` is about commuting coefficients in SNEG, not just numbers evaluable to floating point. A real dispersion relation, a mesh weight and a distribution such as a Dirac delta may all be scalar factors. Declaring an operator-valued function a scalar would allow it to be commuted through its conjugate and silently lose a physical term. Grassmann numbers require their own exchange signs. When defining a linear map F, distinguish $F(zA) = zF(A)$ from the conjugate-linear first slot of an inner product.

B.3 Compatibility and tests

User models should call the public symbols in `Sneg`. Internal helpers in `Sneg`Private`` are implementation details. To qualify an extension, check the single-mode bracket, a three-factor product that can be reduced at either adjacent pair, the adjoint of both results, and a cross-family product. Then specialize to a finite overlap or finite cluster with an independent matrix realization. For example a positive Hermitian overlap matrix G can be realized by $d_i = \sum_j L_{ij} c_j$, $LL^\dagger = G$, using canonical fermions. This checks conjugation and the ordering of the indices as well as signs.

Older interfaces can occur in existing research notebooks. `ZeroMatrix` is an old matrix-utility name; construct a square zero matrix with `ConstantArray[0,{n,n}]`. For package output in a paper use the public `snegtex`; retain `InputForm` alongside formatted output if another program must read index values. Numerical comparisons between SNEG and NRG Ljubljana must use the same impurity and chain-site order, parity convention and zero of energy.

B.4 Interfaces used by existing models

The following names may appear in a model file or in an extension to the algebra. When calling a basis or contraction routine directly, match the state representation and the normalization it expects.

B.4.1 Operators and conventions

`Hubbard` selects the open-chain Hubbard model in `hamiltonian`. It is a model identifier, distinct from the interaction constructor `hubbard`.

`SPININDEXx`, `SPININDEXy`, `SPININDEXz`, `SPININDEXp`, `SPININDEXm` denote component indices 1 through 5 of a spin operator. `antisym[i,j]` encodes the signed remaining Cartesian index; `levicivita` is the tensor form.

`spinflip` changes the last spin bit to one minus the bit. It applies to a single spin-half fermion operator, not to a many-body spin rotation. `outofdiag20` creates a local pair and `outofdiag02` annihilates one; compare the factor order before replacing them with `isospinplus` or `isospinminus`. `isospinx1` and `isospinx2` are the separate pair components of the transverse isospin.

`su3g[i]`, for integer i from 1 to 8, gives the Gell-Mann matrices. For generators t_a normalized by $\mathrm{Tr}(t_a t_b) = \delta_{ab}/2$, use `su3t[i]`, which divides the matrix by two.

`xstates[X]` is the ordered state list of a declared Hubbard-operator family; `xparities[X]` maps state labels to parities. `snegxparity` gives the parity of the transition, the sum of the two state parities modulo two. Make these assignments through `snegxoperators`.

`snegaddop` is the single-family fermion registration step. Use `snegfermionoperators` or `snegspinlessfermionoperators`. `snegnonopQ` marks commuting symbols or function heads. A Grassmann coefficient requires its own declaration, not this marker.

`snegpositiveconstants` also installs rules on `Power` for square roots. For local simplifications use a real-constant declaration and explicit positive assumptions instead.

B.4.2 Symbolic sums and simplification

`snegsumJoinWithRenaming` joins iterator lists when multiplying symbolic sums. `ssJWRnewname` creates and declares an index avoiding the names already in use; `FreeOfIndexQ` tests independence from all indices in a list. Use `nc` on the two sums to include capture avoidance.

`der` supplies derivative rules for combinations containing formal sums. Specify which quantities, including the range and the reference state, are fixed when taking a physical derivative.

`fastpow[op,n]` forms a noncommutative power with intermediate simplifications. `pow` is the direct power constructor. `ruleAbs` replaces a scalar times its conjugate by a squared magnitude.

`SnegSimplifySpin` recognizes spin combinations; `SnegFullSimplify` uses several SNEG transformations within `Simplify`. As with `SnegSimplify`, a recognized constructor can be held for display. Release it before comparing expanded expressions.

`maskOp` evaluates an operation with declared constants temporarily replaced, then restores their names. The temporary symbols do not carry the physical assumptions of the originals.

B.4.3 Contractions and vacuum reduction

`indecesCR` and `indecesAN` locate the creation and annihilation slots of a string; the spelling is historical. `maxcran` bounds the number of contractions. `contract` evaluates a specified list of paired positions, `remainder` gathers the uncontracted operators, and `wickorder` selects terms with a prescribed number of contractions. `wick` includes all orders.

`vevwick2` and `vevwicknew` are alternative evaluation paths with early tests that discard terms; `vevwick` is the general entry point. `Problem` is a diagnostic head in the former path, not an expectation value.

`Zeroonvac` extends `zeroonvac` to additional scalar and formal-sum cases. In the creation-string representation `applytocr[op,state]` also normalizes its nonzero result; it should not be used to measure a transition amplitude.

B.4.4 Symmetry sectors and matrices

`ADD` is a recursive addition step in building a spin multiplet basis. Use `qsbasis` for a complete set of levels. `SU2combine` couples individual highest-weight states with Clebsch-Gordan coefficients; `SU2merge` assembles the resulting sectors. `qscombinefnc` adds the subsystem charges and `qsspincombinefnc` retains the charge of the fermionic subsystem when attaching a spin. Use `SU2basistensorproduct` for the full product.
`Tminus` is the global lowering operator used by the one-argument `transformQStoIS`. Supplying the operator explicitly avoids dependence on an earlier model. `transform1` is one step in removing the descendants from lower-charge sectors.
`mambpair` calculates a transition block for a pair of sectors. Use `makeallmatricesbzvc` or `makeallmatricesbzop` for all connected pairs.
`indexvalue2list` places indexed coefficients into a vector; the ordered basis fixes the meaning of the index.
`snegorthog` orthogonalizes coefficient vectors with a conjugated scalar product. For operator strings use `orthogop` and for occupation vectors `orthogvc`. `pairpattern` tests compatibility of bra and ket slots, including placeholders.
`ZeroMatrix[n]` constructs a square zero matrix. `ConstantArray[0,{n,n}]` is an equivalent dense array.

B.4.5 Utilities and output

`addto` adds an element to a list and removes duplicates. `allcombinations` forms a product of lists of alternatives (`Tuples`); `KS` builds index subsets for the combinatorial routines. `listofpairs` forms ordered pairs drawn from two lists, and `dropzeros` removes literal zeros at the top level.
`TeXString` converts through `TeXForm`; `SnegTeXForm` installs historical formatting rules. Use `snegtex` for a LaTeX string in SNEG operator notation.
`UsageWithMore` assigns a usage message; its old browser-target argument is ignored.
`snegidstring` and `snegcopyright` supply startup text. Read `$SnegVersion` for the package version. `free` is reserved in the public context for compatibility and is not an operator constructor.

C Solutions to the exercises

Tutorial

Solution of Exercise 1.1. At $\epsilon = -U/3$ the empty, spin-doublet, and double energies are $0, -U/3, U/3$. Starting from the doublet, removing costs $U/3$, adding costs $2U/3$. Both excitations change charge and therefore disappear in the restricted calculation. The spin degeneracy costs no energy but does not supply a one-electron transition.

In[1]:=
```
snegfermionoperators[d];
ha = hubbard[d[]] - number[d[]]/3;
en = #[[2,1,1]] & /@ makematricesbzvc[ha, qsbasisvc[{d[]}]];
TableForm[{{First[en]-en[[2]], Last[en]-en[[2]]}},
  TableHeadings -> {None, {"Removal / U", "Addition / U"}}]
```

Out[4]=

Removal / U	Addition / U
$\frac{1}{3}$	$\frac{2}{3}$

Solution of Exercise 2.1. The mixed terms cancel in f^2 and the equal squares enter with opposite signs. In the anticommutator they add. In the number there is an interference term $i\gamma_1\gamma_2$, which is Hermitian because the two Majoranas anticommute. Direct reduction gives the results:

In[5]:=
```
snegmajoranaoperators[gm];
f = (gm[1] + I gm[2])/Sqrt[2];
Expand[{nc[f,f], antikomutator[f,conj[f]], nc[conj[f],f]}]
```

Out[7]= $\{0, 1, \frac{1}{2} + i\mathrm{gm}_1\mathrm{gm}_2\}$

Solution of Exercise 3.1. The molecule has one off-diagonal element $-t$, hence energies $\pm t$. On the two-site ring the two translates contribute to the

same matrix element, giving $\pm 2t$. For $L > 2$ the two neighbours of a site are distinct. Halving the hopping would halve the physical bandwidth.

In[8]:=
```
snegfermionoperators[c];
bm = Select[qszbasisvc[{c[1],c[2]}], First[#] == {-1,1/2} &];
hy = -hop[c[1],c[2]];
Sort[Eigenvalues[#[[1,2]]]] & /@
  (makematricesbzvc[#,bm] & /@ {hy,2 hy}) // MatrixForm
```

Out[11]=
$$\begin{pmatrix} -1 & 1 \\ -2 & 2 \end{pmatrix}$$

Solution of Exercise 4.1. In the inner product the twist factors cancel pointwise. For column indices m, m' the remaining sum is $L^{-1}\sum_{n=0}^{L-1} e^{2\pi i(m-m')n/L} = \delta_{mm'}$. The inverse is the adjoint, not a second forward transform:

In[12]:=
```
ff = Table[Exp[I n (2 Pi j + Pi/2)/3]/Sqrt[3], {n,0,2}, {j,0,2}];
FullSimplify[ConjugateTranspose[ff].ff] // MatrixForm
```

Out[13]=
$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Solution of Exercise 5.1. With $x = e^{-J/T}$ the triplet carries probability $3x/(1+3x)$. Thus $\langle \mathbf{S}_1 \cdot \mathbf{S}_2 \rangle = 3(x-1)/[4(1+3x)]$, tending to $-3/4$ at zero temperature and zero at infinite temperature. Weighting the two energies equally would give a nonzero infinite-temperature result. As a check evaluate the trace of the Boltzmann-weighted spin matrix for $J/T = 1$:

In[14]:=
```
snegspinoperators[{s,1/2}];
eb = Flatten[Eigenvalues[Normal[Last[#]]] & /@
  spinclustermatrices[spinspin[s[1],s[2]], {s[1],s[2]}]];
FullSimplify[(eb.Exp[-eb])/Total[Exp[-eb]] == 3 (1-E)/(4 (E+3))]
```

Out[16]= True

Solution of Exercise 6.1. In the fixed-charge ensemble each spin has probability $1/2$, but the two spins never occur together. Independent half-filled modes have joint probability $1/4$. The difference is the connected two-particle correlator. It can be checked by diagonal matrix elements in the four Fock

states:

In[17]:=
```
bb = qbasisvc[{d[]}];
ddm = makematricesbzvc[hubbard[d[]],bb];
TableForm[{First[First[#]],Tr[Last[#]]/Length[Last[#]]} & /@ ddm,
  TableHeadings -> {None, {"Charge Q","Mean doublons"}}]
```

Out[19]=

Charge Q	Mean doublons
−1	0
0	0
1	1

The $Q = 0$ row gives zero. Including the other sectors with their full dimensions gives $(0 + 0 + 0 + 1)/4$.

Solution of Exercise 7.1. Exactly one of the four local states is doubly occupied. At each site there are 4^{L-1} such configurations; the trace is $UL4^{L-1}$. Hopping has no diagonal element. $\mathrm{Tr}(H_t^2)$ sums squared hopping matrix elements and reveals a missing bond, though not its sign. For two spinful sites of unit hopping:

In[20]:=
```
bb = qbasisvc[{c[1],c[2]}];
mm = Last /@ makematricesbzvc[hy,bb];
{Total[Tr /@ mm], Total[Tr[#.#] & /@ mm]}
```

Out[22]= $\{0, 16\}$

Correlated electrons

The checks below form a fresh session. In each case the calculation uses a different representation or route from the analytical argument.

Solution of Exercise 10.1. At fixed $N = L$, a state with no doublons has one electron on every site. The spins can be chosen independently, giving 2^L ground states. A state at energy U has precisely one doublon and one hole. Choose the doublon site in L ways and the hole site in $L - 1$ ways. There is no spin choice on either of these two sites, giving $L(L-1)2^{L-2}$ states. For four sites the two counts are 16 and 48. The other six half-filled states have two doublons and two holes. Their inclusion gives $\binom{8}{4} = 70$ as required. With U

as the energy unit the full half-filled matrix is diagonal:

In[1]:=
```
Needs["sneg`"];
snegfermionoperators[c, d];
sites4 = c /@ Range[4];
b4 = Select[qbasisvc[sites4], First[#] == {0} &];
atomic4 = First[makematricesbzvc[Total[hubbard[sites4]], b4]][[2]];
TableForm[Sort[Tally[Eigenvalues[atomic4]]],
  TableHeadings -> {None, {"Energy / U", "States"}}]
```

Out[6]=

Energy / U	States
0	16
1	48
2	6

Solution of Exercise 10.2. The singly occupied configurations have zero site energy. The two intermediate doublons cost $U - \delta$ and $U + \delta$. Each is reached from the singlet with squared amplitude $2t^2$, while the triplet cannot hop. Consequently

$$J(\delta) = 2t^2 \left(\frac{1}{U - \delta} + \frac{1}{U + \delta} \right) = \frac{4t^2 U}{U^2 - \delta^2}.$$

Positive excitation costs require $|\delta| < U$, and we need $|t| \ll U - |\delta|$. The divergence as $|\delta| \to U$ signals that a charge state must be retained in the low-energy subspace. It is not a physical divergence of the exchange energy. Verify at finite detuning by diagonalizing the full singlet matrix; the two answers are printed in the same energy units:

In[7]:=
```
snegrealconstants[t, U, delta];
sites2 = {c[1], c[2]};
hd = U Total[hubbard[sites2]] - t hop[c[1], c[2]];
biased = hd + delta (number[c[2]] - number[c[1]])/2;
bs = Select[qsbasisvc[sites2], First[#] == {0, 0} &];
hs = First[makematricesbzvc[biased, bs]][[2]];
par = {U -> 10, delta -> 6, t -> 1/10};
j2 = 4 t^2 U/(U^2 - delta^2);
N[{-Min[Eigenvalues[hs /. par]], j2 /. par}, 6]
```

Out[15]= $\{0.00624172, 0.00625000\}$

The triplet energy is exactly zero even at finite detuning. The difference between the two printed splittings is the contribution of fourth and higher even orders.

Solution of Exercise 10.3. One electron occupies a bonding orbital of energy $-t$. A three-electron state contains exactly one doublon, so its lowest energy is $U - t$. The half-filled ground-state energy is E_- of Eq. (10.2). Substitution in Eq. (10.4) gives

$$\Delta_c = \sqrt{U^2 + 16t^2} - 2t, \qquad \Delta_s = (\sqrt{U^2 + 16t^2} - U)/2.$$

For $U = 0$ both are $2t$: the bonding level forms a closed shell, and the antibonding level lies $2t$ higher. At $U \gg t$, $\Delta_c = U - 2t + 8t^2/U + \cdots$, while $\Delta_s = 4t^2/U + \cdots$. As a direct check, set $(t, U) = (1, 4)$, use charge *only*, and take the lowest eigenvalue separately in each particle-number sector:

In[16]:=
```
bc = qbasisvc[sites2];
mc = makematricesbzvc[hd /. {t -> 1, U -> 4}, bc];
e[q_] := Min[Eigenvalues[First[Select[mc, First[#] == {q} &]][[2]]]];
Simplify[e[1] + e[-1] - 2 e[0] == Sqrt[32] - 2]
```

Out[19]= True

Solution of Exercise 10.4. Expand G_σ in inverse powers of z:

$$G_\sigma(z) = \sum_{\ell \geq 0} M_\ell / z^{\ell+1}.$$

Equation (10.5) then gives $M_\ell = (1 - w)\epsilon^\ell + w(\epsilon + U)^\ell$. Thus

$$M_0 = 1, \quad M_1 = \epsilon + Uw, \quad M_2 = \epsilon^2 + (2\epsilon U + U^2)w.$$

A single pole at the Hartree energy reproduces M_0 and M_1, but has $M_2 = M_1^2$. The exact variance $M_2 - M_1^2 = U^2 w(1 - w)$ vanishes only when the peaks coincide or one carries zero weight. For an independent computation `spectralmoments` evaluates $M_\ell = \langle\{\mathcal{L}^\ell d_\uparrow, d_\uparrow^\dagger\}\rangle$ with $\mathcal{L}A = [A, H]$:

In[20]:=
```
snegrealconstants[ε];
ha = ε number[d[]] + U hubbard[d[]];
mom = spectralmoments[d[AN, UP], d[CR, UP], ha, 2] /.
  avg[number[d[], DO]] -> w;
Expand[mom - {1, ε + U w, ε^2 + (2 ε U + U^2) w}]
```

Out[23]= $\{0, 0, 0\}$

Further applications

Solution of Exercise 8.1. The antisymmetric combination of sites 1 and 2 has energy 2 in units of t. In the remaining subspace the trace is -2 and the product of eigenvalues is -2, giving $-1 \pm \sqrt{3}$. The sum of all energies is zero, as is the sum of the diagonal potential in a magnetic cell. The sum of their squares is $6 + 6 = 12$: six from the on-site potentials and six from oriented links.

In[1]:=
```
hm = {{-2,-1,-1},{-1,1,-1},{-1,-1,1}};
{Eigenvalues[hm], Tr[hm], Tr[hm.hm]}
```

Out[2]= $\{\left\{-1-\sqrt{3}, 2, \sqrt{3}-1\right\}, 0, 12\}$

Solution of Exercise 9.1. Minimize the lower-surface energy in the nonnegative radius r. Its stationary point is $|g|/K$ with energy $-g^2/(2K)$ measured from the undistorted crossing. For $K = M\omega^2$ the coefficient of $b + b^\dagger$ is $g/\sqrt{2M\omega}$; the square divided by ω is precisely the relaxation energy. Diagonalize the electronic part on a circle with the same radius in two different directions:

In[3]:=
```
el[x_,y_] := x PauliZ + y PauliX;
Sort[Eigenvalues[el[3,4]]] == Sort[Eigenvalues[el[0,5]]]
```

Out[4]= True

Solution of Exercise 11.1. At particle–hole symmetry both diagonal even energies vanish. Pair transfer splits this subspace into levels $-\Gamma, \Gamma$, while each odd state is at $-U/2$. The doublet wins for $U > 2\Gamma$; at equality it is degenerate with the lower singlet. Test the crossing at $\Gamma = 1$ without imposing charge

symmetry:

In[5]:=
```
snegfermionoperators[d];
hp = 2 hubbard[d[]] - number[d[]] - bcs[d[]];
mm = makematricesbzvc[hp, nonebasisvc[{d[]}]];
Sort[Eigenvalues[First[mm][[2]]]]
```

Out[8]= $\{-1,-1,-1,1\}$

Solution of Exercise 12.1. Addition of three spins $1/2$ gives one $S = 3/2$ quartet and two $S = 1/2$ doublets. Their energies are $3J/4$ and $-3J/4$, respectively. Hence $Z = 8\cosh(3J/4T)$ and the residual entropy is $\ln 4$ in units of k_B. No choice of quantization axis changes that count:

In[9]:=
```
snegspinoperators[{s,1/2}];
tri = Total[spinspin @@@ Subsets[s /@ Range[3], {2}]];
ev = Flatten[Eigenvalues[Normal[Last[#]]] & /@
  spinclustermatrices[tri,{s -> Range[3]}]];
TableForm[Sort[Tally[ev]], TableHeadings -> {None, {"Energy / J","
    States"}}]
```

Out[12]=

Energy / J	States
$-\frac{3}{4}$	4
$\frac{3}{4}$	4

Solution of Exercise 13.1. Here $G_{dd}(z) = z/(z^2 - V^2)$: each of the bonding and antibonding orbitals has impurity probability $1/2$. The two half-weight poles merge to the original unit-weight pole as $V \to 0$. This limit is not obtained by discarding one of the two levels.

In[13]:=
```
snegfermionoperators[c];
snegrealconstants[V];
Apart[greensfunction[d[AN,UP],d[CR,UP],V hop[c[],d[]],z],z]
```

Out[15]= $\dfrac{1}{2(z-V)} + \dfrac{1}{2(V+z)}$

Solution of Exercise 14.1. The occupation commutes with the free Hamiltonian, so its retarded commutator vanishes identically. At thermal equilibrium $\partial_\epsilon f = -f(1-f)/T$; changing the level energy changes the Gibbs weights. For

an isolated system a commuting probe cannot redistribute them dynamically. At $T = 1, \epsilon = 0$ check the derivative by a centred difference:

In[16]:=
```
occ[e_] := thermalaverage[number[d[],UP], e number[d[]], {d}, 1];
{komutator[number[d[],UP],number[d[]]],
 Round[(occ[1/1000]-occ[-1/1000])/(2/1000),1/10000]}
```

Out[17]= $\{0, -\frac{1}{4}\}$

Bibliography

[1] R. Žitko, "SNEG – Mathematica package for symbolic calculations with second-quantization-operator expressions", Comput. Phys. Commun. **182**, 2259–2264 (2011).

[2] J. Vučičević and R. Žitko, "Electrical conductivity in the Hubbard model: orbital effects of magnetic field", Phys. Rev. B **104**, 205101 (2021).

[3] J. Hubbard, "Electron correlations in narrow energy bands", Proc. R. Soc. London A **276**, 238–257 (1963).

[4] W.-H. Steeb and Y. Hardy, *Bose, spin and fermi systems: problems and solutions* (World Scientific, 2015).

[5] A. H. MacDonald, S. M. Girvin, and D. Yoshioka, "t/U Expansion for the Hubbard model", Phys. Rev. B **37**, 9753–9756 (1988).

[6] A. Auerbach, *Interacting electrons and quantum magnetism* (Springer, 1994).

[7] Á. Bácsi, T. Iličin, and R. Žitko, "Systematic schrieffer-wolff transformation approach to josephson junctions: quasiparticle effects and josephson harmonics", Phys. Rev. B **113**, 064501 (2026).

[8] A. Horvat, R. Žitko, and J. Mravlje, "Low-energy physics of three-orbital impurity model with Kanamori interaction", Phys. Rev. B **94**, 165140 (2016).

[9] T. Iličin and R. Žitko, "Variational solution of the superconducting Anderson impurity model and the band-edge singularity phenomena", SciPost Phys. **19**, 006 (2025).

[10] T. Iličin and R. Žitko, *Quasiparticle-resolved variational theory of Andreev spin qubits*, (2026) https://arxiv.org/abs/2610.08557.

[11] L. Pavešić, R. Aguado, and R. Žitko, "Strong-coupling theory of quantum-dot Josephson junctions: role of a residual quasiparticle", Phys. Rev. B **109**, 125131 (2024).

[12] L. Pavešić, M. Pita-Vidal, A. Bargerbos, and R. Žitko, "Impurity Knight shift in quantum dot Josephson junctions", SciPost Phys. **15**, 070 (2023).

[13] R. Žitko, R. Peters, and T. Pruschke, "Properties of anisotropic magnetic impurities on surfaces", Phys. Rev. B **78**, 224404 (2008).

Index of functions

$SnegVersion....246
abrikosov....125
acmt....246
ADD....266
addto....266
allcombinations....266
AN....78
anhop....105
anomaloushop....106
anticommutator....93
antikomutator....93
antisym....264
ap....170
appendquantumnumber....171
applybasis....171
applytocr....265
asciitosneg....243
avg....225
basis....172
BASIS....172
basistensorproduct....172
bcs....106
bdghamiltonian....226
blochhamiltonian....148
blochphase....149
blochphaseexp....150
block....94
bogoliubov....227
bogoliubovrules....229
bosonQ....247
bra....173
braket....173
braketop....220
braketrule....174
braketvc....174
bzop2bzvc....175
bzQ....175
bzvc2bzop....176
chargecharge....107
clusterhop....150
cmt....247
coh....78
commutator....94
conj....94
contract....265
contraction....220
CR....79
current....108
currentphi....108
dd....221
decomposeop....176
decomposevc....176
der....265
direct....177
directproductbasis....177
DO....79
dropemptysubspaces....177
dropzeros....266
eigenoperators....237
EMPTY....95
eomclosure....240
expvop....221
expvvc....178
FactorSeparableSumCustom....151

fastpow 265
fermionQ 248
fermionstox 126
fixspin 178
fixspinket 255
fourier 151
fourierrules 153
free 266
FreeOfIndexQ 265
FullSimplifyKD 154
gaussianaverage 230
genanhop 109
genhop 109
goodopQ 179
grassmanQ 248
greensfunction 240
halfintegerQ 80
hamiltonian 110
holehop 111
holsteinprimakoff 126
hop 111
hopphi 112
hubbard 113
Hubbard 264
indecesAN 265
indecesCR 265
indexvalue2list 266
inner 95
int 80
interleave 179
invertspin 127
isannihilation 249
iscreation 249
isnumericQ 250
isospin 127
isospinminus 128
isospinplus 129
isospinx 129
isospinx1 264
isospinx2 264
isospinxyz 129
isospiny 130
isospinz 130
isozsq 113
jordanwigner 130
ket 179
ketbratensorproduct 180
keymerge 181
keyselect 181
komutator 96
KS 266
latticeindexQ 154
latticeindextype 154
latticesize 155
lehmann 241
levicivita 131
liouvillian 242
listbosonoperators 80
listcomplexconstants 81
listfermionoperators 81
listfreeindexes 82
listgrassmanconstants 82
listintegerconstants 82
listlatticeindexes 155
listlatticevectors 155
listmajoranaoperators 83
listmomentumindexes 155
listofpairs 266
listrealconstants 83
listspincomponentindexes 255
listspinoperators 83
listxoperators 84
lrmap 181
majoranaQ 251
makeallmatricesbzop 182
makeallmatricesbzvc 182
makebasis 183
makematricesbzop 184
makematricesbzvc 184
mambpair 266

manyisospin....132
manyspin....132
mapbasis....186
maskOp....265
matrixhop....114
matrixrepresentationop....186
matrixrepresentationvc....187
matrixrepresentationvcfast....187
matrixrepresentationvcsparse....188
maxcran....265
mcross....96
meanfield....231
meanfieldsolve....232
mergebasis....189
momentumbasis....156
momentumindexQ....156
nambu....115
nc....97
ndx2op....189
nnop....189
NONE....98
nonebasis....190
nonebasisvc....190
normalorder....98
normalorderwick....222
normop....190
normvc....191
number....115
op2ndx....191
operatorQ....251
ops2vc....191
ordering....99
orthogbasisop....192
orthogbasisvc....192
orthogop....193
orthogvc....193
outer....100
outofdiag02....264
outofdiag20....264
pairpattern....266
PauliMinus....132
PauliPlus....133
PauliX....133
PauliY....133
PauliZ....134
phononbasis....194
phononid....194
phononminus....194
phononnumber....195
phononplus....195
phononx....195
pow....101
PrettyInput....101
PrettyOutput....244
Problem....265
PROJ0....116
PROJ02....116
PROJ1....117
PROJ2....117
PROJDO....117
projector....117
projector0....118
projector02....119
projector1....119
projector2....119
projectorDO....120
projectorE....120
projectorEphi....121
projectorO....121
projectorOphi....122
projectorUP....122
PROJUP....122
qbasis....196
qbasisvc....196
qsbasis....196
qsbasisvc....197
qscombinefnc....266
qsspincombinefnc....266
QSZaddspinket....198
qszbasis....199

qszbasisvc 199
quadraticform 234
quadraticoperator 236
quickDBL 200
quickDBLSZ 201
quickISObasis 201
quickISOSZbasis 202
quickSU2basis 202
remainder 265
remove1state 203
removestates 203
rotatespin 134
ruleAbs 265
ruleExpandUnderSum 156
ruleFullSimplifyUnderSum 157
rulesimplifyscalar 256
rulestripscalar 157
rulesumDistribute 158
rulesumFactor 158
sbasis 204
sbasisvc 204
scalar 251
scalarproduct 205
scalarproductop 205
scalarproductvc 205
schriefferwolff 238
schriefferwolffmatrix 239
schwingerbosons 135
SEA 101
showbzmat 206
SimplifyKD 158
snegaddop 264
snegAssuming 102
snegbosonoperators 84
snegcomplexconstants 85
snegcomplexfunctions 85
snegcopyright 266
snegfermionoperators 85
snegfreeindexes 86
SnegFullSimplify 265
sneggrassmanconstants 87
snegidstring 266
snegintegerconstants 87
sneglatticeindexes 159
sneglatticevectors 160
sneglinearoperator 252
sneglinearoperatorFirst 252
snegmajoranaoperators 88
snegmomentumindexes 160
snegnonopQ 264
snegold2newrules 244
snegOrderedQ 253
snegorthog 266
snegpositiveconstants 264
snegrealconstants 88
snegrealfunctions 89
snegSeries 102
snegsesquilinearoperator 254
SnegSimplify 103
SnegSimplifySpin 265
snegspincomponentindexes 135
snegspinlessfermionoperators 89
snegspinoperators 90
snegsumJoinWithRenaming 265
snegtex 245
SnegTeXForm 266
snegtoascii 246
snegxoperators 91
snegxparity 264
spectralmoments 242
spinall 136
spinbasis 206
spinbra 207
spinclustermatrices 136
spincomponentindexQ 137
spincomponents 137
spindown 138
spindownvc 207
spinflip 264
spinfliphop 123

spinfliphopphi 123
SPININDEXm 264
SPININDEXp 264
SPININDEXx 264
SPININDEXy 264
SPININDEXz 264
spinket 208
spinketbraI 208
spinketbraM 256
spinketbraOPS 256
spinketbraP 257
spinketbrarules 208
spinketbraspinspin 209
spinketbraX 209
spinketbraXYZ 257
spinketbraY 257
spinketbraZ 257
spinlessbasis 210
spinmatrixM 258
spinmatrixP 258
spinmatrixX 138
spinmatrixY 258
spinmatrixZ 259
spinminus 139
spinof 91
spinplus 139
spinQ 255
spinspin 123
spinspinmp 140
spinspinpm 140
spinspinsymmetric 124
spinspinx 140
spinspinxy 141
spinspiny 141
spinspinz 141
spinss 142
spintensor 142
spinx 143
spinxyz 143
spinxyzgen 144
spiny 144
spinz 145
ssJWRnewname 265
stevens 145
SU2basistensorproduct 210
SU2combine 266
SU2merge 266
su3g 264
su3t 264
sum 161
sum2list 162
sumAbstractIndex 162
sumAutoRename 163
sumCluster 164
sumCollect 164
sumCollectSimplifyKD 165
sumExpand 165
sumFullSimplify 166
sumMomentumCanonical 166
sumNameIndex 167
sumShift 168
sumSimplify 168
sumSimplifyKD 169
sumThread 170
superanticommutator 103
superantikomutator 104
supercommutator 104
superkomutator 104
szbasis 211
szbasisvc 211
TeXString 266
thermalaverage 243
Tminus 266
transform1 266
transformbasis 212
transformfunc 212
transformQStoIS 213
transformtoLR 214
transformtoLRvc 215
transformtoPH 215

twohop 125
UP 92
UsageWithMore 266
VACUUM 92
vacuum 216
vc 217
vc2ops 218
vccolor 218
vccolorDefault 219
vccolorSimple 219
vev 222
vevwick 223
vevwick2 265
vevwicknew 265
VMV 105
wick 224
wickorder 265
xoperator 146
xoperatorQ 93
xparities 264
xstates 264
xtofermions 147
xtospin 147
ZeroMatrix 266
zeroonvac 219
Zeroonvac 265