Microsoft.Research.Liquid Namespace

▲ Classes

	Class	Description
* \$	Арр	Utilities for executing the Liquid application.
4 3	Bit	Represents the measured value, in the computational basis, of a qubit.
4 3	Circuit	The circuit representation of an operation in a quantum algorithm. Circuits are generally created using Circuit.Compile.
? \$	CMat	A dense matrix of complex numbers.
? \$	CSMat	A sparse matrix of complex numbers.
43	CVec	A block-sparse vector of complex numbers.
<i>≪≴</i>	Fermion	Hamiltonian simulation for fermionic systems.
e		

	Gate	A quantum gate.
4 3	GateOp	Gate operation type. This is used in Gate definitions.
* \$	GrowPars	Parameters that control circuit growth. See Circuit.Grow, Circuit.GrowGates, and Circuit.GrowSingle.
4 3	Hamiltonian	Base class for Hamiltonian dynamics simulators.
9 \$	HamiltonianGates	A collection of gates that are useful for Hamiltonian simulation and annealing.
? \$	Ket	Represents a state vector.
<i>9</i> 3	KrausOp	Entries for Kraus operators in Channel Gate type
4 3	Noise	A complete noise model for a specific circuit.
€ \$	NoiseEvents	Noise statistics that are tracked for normal and error-correcting gates.
43	NoiseModel	A noise model for a particular type of gate (or set of gates).
4 3	NoiseStat	Statistics tracked for each time that noise is applied.
4 \$	NoisyMats	Utility class for computing a Pauli rotation matrix. This is used to run quantum

		chemistry circuits with noise injected.
*\$	Operations	The Operations module provides definitions of basic gates. It also includes some handy operators for manipulating qubit lists, and some operations for building gates from existing gates.
4 3	QECC	Base class for quantum error correcting codes.
4 3	Qubit	Represents a quantum bit. New Qubits are created using the Ket Add methods.
43	RunMode	Trotterization types.
4 3	Spin	Hamiltonian for spin systems, such as the Ising model or a spin glass.
*\$	SpinTerm	A single term in a Spin Hamiltonian.
*\$	Stabilizer	A stabilizer-based simulator based on CHP by Scott Aaronson and Daniel Gottesman. See arXiv:quant- ph/0406196 for more details.
*\$	Steane7	Implementation of a Steane 7- bit quantum error correcting code, [[7,1,3]], based on the QECC class.

	* \$	Tests Util		A collection of sample Liquid simulations and tests, plus some utility routines to make it easier to write new samples.
	* \$			General utilities used by the rest of the system
	* UtilLQDAttribute * UtilprocStatsT		oute	Allows a function to be visable from a LIQUiD script or the command line
			sТ	Current process memory usage statistics. Returned by the procStates function.
4	Structur	es		
		Structure	Desc	ription
		Complex Data		type for complex numbers.

App Class

Utilities for executing the Liquid application.

Inheritance Hierarchy

SystemObject Microsoft.Research.LiquidApp

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

```
[<AbstractClassAttribute>]
[<SealedAttribute>]
type App = class end
```

The App type exposes the following members.

Methods

	Name	Description
= ≬ S	RunLiquid	The main routine for running Liquid. This function gets the command-line parameters from the environment, parses them, and executes the requested Liquid method.

Тор

⊿ See Also

Reference

Microsoft.Research.Liquid Namespace

App Methods

The App type exposes the following members.

Methods

	Name	Description
≕∳ S	RunLiquid	The main routine for running Liquid. This function gets the command-line parameters from the environment, parses them, and executes the requested Liquid method.

Тор

⊿ See Also

Reference App Class Microsoft.Research.Liquid Namespace

AppRunLiquid Method

The main routine for running Liquid. This function gets the commandline parameters from the environment, parses them, and executes the requested Liquid method.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____ Copy _____ Static member RunLiquid : unit -> int

Return Value Type: Int32 0 if execution succeeded, or 1 if an error occured.

⊿ See Also

Reference App Class Microsoft.Research.Liquid Namespace

Bit Class

Represents the measured value, in the computational basis, of a qubit.

▲ Inheritance Hierarchy

SystemObject Microsoft.Research.LiquidBit

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#		Сору			
[<sealedattribute>]</sealedattribute>					
[<s< td=""><td>erializableAttribute>]</td><td></td></s<>	erializableAttribute>]				
typ	e Bit =				
	class				
	<pre>interface IEquatable<bit></bit></pre>				
	<pre>interface IStructuralEquatable</pre>				
	<pre>interface IComparable<bit></bit></pre>				
	interface IComparable				
	interface IStructuralComparable				
	end				

The Bit type exposes the following members.

Properties

Name	Description
V	The integer value of a measured qubit, either 0 or 1. Note that this property will throw an exception if the value is

unknown.

Тор

Methods

	Name	Description
≡Q	Dump	Dumps the measured value.
=♥	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
=♥	GetHashCode	Serves as the default hash function. (Inherited from Object.)
≓∳	GetType	Gets the Type of the current instance. (Inherited from Object.)
=♥	ToString	Gets a string representation of this value. The string will be "Zero", "One", or "?". (Overrides ObjectToString.)

Тор

Remarks

Possible values are:

- **Unknown**: The qubit has not been measured since it was initialized or since the last time it was reanimated.
- **Zero**: The qubit was last measured as |0> in the computational basis and has not been reanimated since.
- One: The qubit was last measured as |1> in the computational

basis and has not been reanimated since.

⊿ See Also

Reference Microsoft.Research.Liquid Namespace

Bit Properties

The Bit type exposes the following members.

▲ Properties

◢

	Name	Description
*	V	The integer value of a measured qubit, either 0 or 1. Note that this property will throw an exception if the value is unknown.
Тор		
See A	SO	

Reference Bit Class Microsoft.Research.Liquid Namespace

Bitv Property

The integer value of a measured qubit, either 0 or 1. Note that this property will throw an exception if the value is unknown.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



⊿ See Also

Reference Bit Class Microsoft.Research.Liquid Namespace

Bit Methods

The Bit type exposes the following members.

Methods

	Name	Description
=∳	Dump	Dumps the measured value.
=♥	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
=∲	GetHashCode	Serves as the default hash function. (Inherited from Object.)
=∳	GetType	Gets the Type of the current instance. (Inherited from Object.)
≡♥	ToString	Gets a string representation of this value. The string will be "Zero", "One", or "?". (Overrides ObjectToString.)

Тор

⊿ See Also

Reference Bit Class Microsoft.Research.Liquid Namespace

BitDump Method

Dumps the measured value.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

f

Type: Microsoft.FSharp.CoreFSharpOptionFSharpFuncInt32, FSharpFuncString, Unit

The optional output function to use. The default is showLogInd. *level*

Type: Microsoft.FSharp.CoreFSharpOptionInt32

The optional indentation level. The default is 0.

⊿ See Also

```
Reference
Bit Class
Microsoft.Research.Liquid Namespace
```

BitToString Method

Gets a string representation of this value. The string will be "Zero", "One", or "?".

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

_ Copy _

abstract ToString : unit -> string
override ToString : unit -> string

Return Value Type: String The string

⊿ See Also

Reference Bit Class Microsoft.Research.Liquid Namespace

Circuit Class

The circuit representation of an operation in a quantum algorithm. Circuits are generally created using Circuit.Compile.

▲ Inheritance Hierarchy

```
SystemObject Microsoft.Research.LiquidCircuit
```

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



The Circuit type exposes the following members.

▲ Properties

		Name	Description		
	*	NotEmpty	Is this Circuit element not the "Empt circuit.		
	Тор				
⊿ Methods					
		Name	D	Description	
	= ≬ S	Compile	С	Compiles a function	

		implemented as a sequence of gate function calls into a Circuit.
≓Ŵ	Dump	Dumps this circuit recursively to the console and/or log.
≓Ŵ	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
= 0	FindIds	Find Ids below this point and the time needed to execute the circuit (internal)
= 0	Fold	Rewrites this circuit by identifying opportunities for parallelism and turning Seq elements into Par elements where possible. This makes for a better rendering.
≡≬	GateCount	Gets the count of gates in circuit. Note that pure label gates that is, gates of type String are never counted.
≓∳	GetHashCode	Serves as the default hash function. (Inherited from Object.)
≓∳	GetType	Gets the Type of the current instance. (Inherited from Object.)
≓Ŵ	Grow	Creates an optimized "grown" version of this Circuit. The type and limits of the conversion are

		specified in the grow parameter.
≡♥	GrowGates	Creates an equivalent circuit to this circuit by aggregating existing unitary gates into larger unitary gates. The new circuit will execute faster because it has fewer matrix applications required.
.≡♥	GrowSingle	Converts this circuit into a single unitary gate, implemented by a single matrix. This allows maximum optimization of circuit execution. This circuit must be completely unitary to be converted into a single matrix. If it is not, an exception will be raised by this method.
=\$	RemoveRedund	Creates a new circuit logically equivalent to this circuit, but with redundant gates removed. For instance, if this routine found a sequence of two X gates in succession on the same qubit, it would remove both of them from the result.
-= Q	Render	Renders this circuit to a file.
≡ ∲	RenderHT	Renders a circuit to both svg, for HTML, and tikz, for TeX.
≓Ŷ	Reverse	Reverses this circuit, which must contain only unitary gates. In addition, bottom-level gates are replaced by their adjoints.

		This method will raise an exception if this circuit contains a non-unitary gate.
=♥	Run	Runs this Circuit. The state of the Ket containing the passed-in qubits will be modified.
≓∳	ToString	Returns a string representing the current circuit element. (Overrides ObjectToString.)
= 0	Wires	Gets the list of wires (qubit IDs) touched by this circuit.

Тор

Remarks

The possible types of circuits are:

- Seq: A list of sub-circuits executed equentially.
- Par: A list of sub-circuits executed in parallel.
- **Apply**: Application of a single gate.
- Ext: Application of a single gate that extends a parent, such as an adjoint or control. This type of circuit is created from a Modify gate.
- **BitCon**: Application of a classically-controlled (BCOp) gate.
- Wrap: A sub-circuit wrapped into a single gate for more concise rendering.
- **Empty**: A circuit that does nothing.

⊿ See Also

Reference

Microsoft.Research.Liquid Namespace

Circuit Properties

The Circuit type exposes the following members.

▲ Properties

		Name	Description
		NotEmpty	Is this Circuit element not the "Empty" circuit.
	Тор		
▲ See Also			
	Reference Circuit Class Microsoft.Res	search.Liqui	d Namespace

CircuitNotEmpty Property

Is this Circuit element not the "Empty" circuit.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору ____

member NotEmpty : bool with get

Property Value Type: Boolean

⊿ See Also

Reference Circuit Class Microsoft.Research.Liquid Namespace

Circuit Methods

The Circuit type exposes the following members.

Methods

	Name	Description
=\$ S	Compile	Compiles a function implemented as a sequence of gate function calls into a Circuit.
≓ ©	Dump	Dumps this circuit recursively to the console and/or log.
= 0	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
=♥	FindIds	Find Ids below this point and the time needed to execute the circuit (internal)
≓ ∳	Fold	Rewrites this circuit by identifying opportunities for parallelism and turning Seq elements into Par elements where possible. This makes for a better rendering.
≡ ⊘	GateCount	Gets the count of gates in circuit. Note that pure label gates that is, gates of type String are never counted.

≓Ŵ	GetHashCode	Serves as the default hash function. (Inherited from Object.)
=\$	GetType	Gets the Type of the current instance. (Inherited from Object.)
≓Ŵ	Grow	Creates an optimized "grown" version of this Circuit. The type and limits of the conversion are specified in the grow parameter.
₩	GrowGates	Creates an equivalent circuit to this circuit by aggregating existing unitary gates into larger unitary gates. The new circuit will execute faster because it has fewer matrix applications required.
≡ ⊗	GrowSingle	Converts this circuit into a single unitary gate, implemented by a single matrix. This allows maximum optimization of circuit execution. This circuit must be completely unitary to be converted into a single matrix. If it is not, an exception will be raised by this method.
≓Ŵ	RemoveRedund	Creates a new circuit logically equivalent to this circuit, but with redundant gates removed. For instance, if this routine found a sequence of two X gates in succession on the

		same qubit, it would remove both of them from the result.
=	Render	Renders this circuit to a file.
≡Ŵ	RenderHT	Renders a circuit to both svg, for HTML, and tikz, for TeX.
-	Reverse	Reverses this circuit, which must contain only unitary gates. In addition, bottom-level gates are replaced by their adjoints. This method will raise an exception if this circuit contains a non-unitary gate.
= Q	Run	Runs this Circuit. The state of the Ket containing the passed-in qubits will be modified.
-=	ToString	Returns a string representing the current circuit element. (Overrides ObjectToString.)
= Q	Wires	Gets the list of wires (qubit IDs) touched by this circuit.

Тор

⊿ See Also

Reference Circuit Class Microsoft.Research.Liquid Namespace

CircuitCompile Method

Compiles a function implemented as a sequence of gate function calls into a Circuit.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

f

Type: Microsoft.FSharp.CoreFSharpFuncFSharpListQubit, Unit

The gate function to compile.

qs

Type: **Microsoft.FSharp.CollectionsFSharpListQubit** The qubits the new Circuit will operate on.

Return Value

Type: Circuit A new Circuit that represents the function calls

⊿ See Also

Reference Circuit Class Microsoft.Research.Liquid Namespace

CircuitDump Method

Dumps this circuit recursively to the console and/or log.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

f

Type: Microsoft.FSharp.CoreFSharpOptionFSharpFuncInt32, FSharpFuncString, Unit

The optional output function to use. The default is showLogInd. *level*

Type: **Microsoft.FSharp.CoreFSharpOptionInt32** The optional indentation level. The default is 0.

⊿ See Also

Reference

Circuit Class Microsoft.Research.Liquid Namespace

CircuitFindIds Method

Find Ids below this point and the time needed to execute the circuit (internal)

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

detail

Type: SystemInt32

What level of detail for when we hit a "Wrap" gate *cntEmpty*

Type: Microsoft.FSharp.CoreFSharpOptionBoolean

An option allowing Empty gates to be counted like Native The default is false, which is to not count Empty gates as Native.

Return Value

Type: Tuple**FSharpSet**Int32, Int32 Set of ids,total time to execute

⊿ See Also

Reference Circuit Class Microsoft.Research.Liquid Namespace

CircuitFold Method

Rewrites this circuit by identifying opportunities for parallelism and turning Seq elements into Par elements where possible. This makes for a better rendering.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#	Сору
member Fold : agressive : FSharpOption <bool> -> Cir</bool>	cuit
<[]	

Parameters

agressive

Type: Microsoft.FSharp.CoreFSharpOptionBoolean An option to fold as much as possible by decomposing this circuit into basic gates before folding. The default is to not decompose Wrap gates and other aggregates before folding.

Return Value Type: Circuit The new folded circuit

⊿ See Also

Reference Circuit Class Microsoft.Research.Liquid Namespace

CircuitGateCount Method

Gets the count of gates in circuit. Note that pure label gates -- that is, gates of type String -- are never counted.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

doParallel

Type: Microsoft.FSharp.CoreFSharpOptionBoolean

An option to only count the longest sub-circuit of a parallel component, rather than adding all gates in all components. This is useful when computing gate depth. The default is false, which counts all gates, summing over parallel sub-circuits.

gMatch

Type: Microsoft.FSharp.CoreFSharpOptionFSharpFuncGate, Boolean

An optional function to filter which gates should be counted. If a match function is provided, only gates that return true are included in the count. The default is to count all non-String gates.

Return Value

Type: Int32

The count of low-level gates in the circuit

⊿ See Also

Reference Circuit Class Microsoft.Research.Liquid Namespace

CircuitGrow Method

Creates an optimized "grown" version of this Circuit. The type and limits of the conversion are specified in the grow parameter.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# Copy
member Grow :
    k : Ket *
    gp : GrowPars -> Circuit
```

Parameters

k

Type: Microsoft.Research.LiquidKet A Ket this Circuit could be run with.

gр

Type: Microsoft.Research.LiquidGrowPars The grow parameters to use. See the GrowPars type for details.

Return Value Type: Circuit A new, optimized Circuit

⊿ See Also

Reference Circuit Class Microsoft.Research.Liquid Namespace
CircuitGrowGates Method

Creates an equivalent circuit to this circuit by aggregating existing unitary gates into larger unitary gates. The new circuit will execute faster because it has fewer matrix applications required.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

k

Type: Microsoft.Research.LiquidKet A Ket this Circuit could be run with.

gp

Type: **Microsoft.FSharp.CoreFSharpOptionGrowPars** Optional grow parameters. The default is to use all of the GrowPars defaults. See the GrowPars type for details.

Return Value Type: Circuit New optimized Circuit

▲ See Also

Reference Circuit Class Microsoft.Research.Liquid Namespace

CircuitGrowSingle Method

Converts this circuit into a single unitary gate, implemented by a single matrix. This allows maximum optimization of circuit execution. This circuit must be completely unitary to be converted into a single matrix. If it is not, an exception will be raised by this method.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

Parameters

gр

Type: Microsoft.Research.LiquidGrowPars Optional grow parameters. The default is to use all of the GrowPars defaults. See the GrowPars type for details.

Return Value Type: Circuit The new single-gate Circuit

⊿ See Also

Reference Circuit Class Microsoft.Research.Liquid Namespace

CircuitRemoveRedund Method

Creates a new circuit logically equivalent to this circuit, but with redundant gates removed. For instance, if this routine found a sequence of two X gates in succession on the same qubit, it would remove both of them from the result.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

member RemoveRedund : unit -> Circuit

Return Value Type: Circuit The new, trimmed circuit

⊿ See Also

Reference Circuit Class Microsoft.Research.Liquid Namespace

CircuitRender Method

Renders this circuit to a file.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#	Сору
member Render :	
<i>file</i> : string *	
<i>typ</i> : FSharpOption <string> *</string>	
<pre>detail : FSharpOption<int> *</int></pre>	
<pre>split : FSharpOption<float> *</float></pre>	
<pre>scale : FSharpOption<float> -> unit</float></pre>	

Parameters

file

Type: SystemString

The full name of the file to create, including the extension.

typ

Type: Microsoft.FSharp.CoreFSharpOptionString

The optional format for the rendered graphics. Possible values are:

- "qc": QCircuit Liquid format
- "tikz": TikZ Liquid format for TeX and LaTeX
- "svg": Vector graphics format for HTML

The default is "svg".

detail

Type: Microsoft.FSharp.CoreFSharpOptionInt32

An option specifying how many levels of Wrap to unwrap. The default is 999.

split

Type: Microsoft.FSharp.CoreFSharpOptionDouble

An option specifying what percentage of the total gates should go into each figure, if the circuit won't fit into a single figure. The default value varies with the figure size.

scale

Type: Microsoft.FSharp.CoreFSharpOptionDouble

An option specifying a scaling percentage for the rendering, with 100.0 being full size. The default value varies with the figure size.

Remarks

Detail is 0 for least, increasing wrap levels by each increment. That is, at level 0 only the top-level gate of a Wrap is displayed; at level 1, the top-level circuit that implements the Wrap is displayed; at l;evel 2, the top-level circuit is displayed with its Wrap gates also unwrapped once; etc.

⊿ See Also

Reference Circuit Class Microsoft.Research.Liquid Namespace

CircuitRenderHT Method

Renders a circuit to both svg, for HTML, and tikz, for TeX.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#
 Copy
member RenderHT :
 file : string *
 detail : FSharpOption<int> *
 split : FSharpOption<float> *
 scale : FSharpOption<float> -> unit

Parameters

file

Type: SystemString

The base name of the files to create. Appropriate extensions will be added for the created files: ".htm" for the SVG rendering, and ".tex" for the TikZ rendering.

detail

Type: Microsoft.FSharp.CoreFSharpOptionInt32

An option specifying how many levels of Wrap to unwrap. The default is 999.

split

Type: Microsoft.FSharp.CoreFSharpOptionDouble

An option specifying what percentage of the total gates should go into each figure, if the circuit won't fit into a single figure. The default value varies with the figure size.

scale

Type: Microsoft.FSharp.CoreFSharpOptionDouble

An option specifying a scaling percentage for the rendering, with 100.0 being full size. The default value varies with the figure size.

Remarks

Detail is 0 for least, increasing wrap levels by each increment. That is, at level 0 only the top-level gate of a Wrap is displayed; at level 1, the top-level circuit that implements the Wrap is displayed; at l;evel 2, the top-level circuit is displayed with its Wrap gates also unwrapped once; etc.

⊿ See Also

Reference Circuit Class Microsoft.Research.Liquid Namespace CircuitRender(String, FSharpOptionString, FSharpOptionInt32, FSharpOptionDouble, FSharpOptionDouble)

CircuitReverse Method

Reverses this circuit, which must contain only unitary gates. In addition, bottom-level gates are replaced by their adjoints. This method will raise an exception if this circuit contains a non-unitary gate.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____ Copy _____ Copy

Return Value Type: Circuit

The resulting reversed circuit

⊿ See Also

Reference Circuit Class Microsoft.Research.Liquid Namespace

CircuitRun Method

Runs this Circuit. The state of the Ket containing the passed-in qubits will be modified.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

qs

Type: **Microsoft.FSharp.CollectionsFSharpListQubit** The list of Qubits to operate on. These must correspond to the qubits the circuit was compiled with; that is, they must have the same qubit IDs as those.

⊿ See Also

Reference Circuit Class Microsoft.Research.Liquid Namespace

CircuitToString Method

Returns a string representing the current circuit element.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору _

```
abstract ToString : unit -> string
override ToString : unit -> string
```

Return Value Type: String The string representation

⊿ See Also

Reference Circuit Class Microsoft.Research.Liquid Namespace

CircuitWires Method

Gets the list of wires (qubit IDs) touched by this circuit.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Copy _

member Wires : unit -> FSharpList<int>

Return Value Type: **FSharpListInt32** A list of the wires used

⊿ See Also

Reference Circuit Class Microsoft.Research.Liquid Namespace

CMat Class

A dense matrix of complex numbers.

Inheritance Hierarchy

```
SystemObject Microsoft.Research.LiquidCMat
```

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

⊿ Syntax

F#

```
[<SerializableAttribute>]
type CMat = class end
```

The CMat type exposes the following members.

Constructors

	Name	Description
. ≡ \$	CMat(Double, Double)	Creates a new matrix from arrays of real and imaginary parts.
≓ Q	CMat(Int32, FSharpListTupleInt32, Int32, Double, Double)	Creates a square matrx from a sparse list of elements.
=∳	CMat(Int32, FSharpOptionBoolean)	Creates a square identity or zero matrix.

Тор

▲ Properties

	Name	Description
*	Item	Gets an individual element of the matrix, as a Complex number.
*	Length	The number of rows or columns in the matrix, for square matrices.
	LengthC	The number of columns in the matrix.
	LengthR	The number of rows in the matrix.

Тор

▲ Methods

	Name	Description
-	Add	Adds another matrix to this matrix. The two matrices must have the same number of rows and columns.
-= Q	Adj	Computes the adjoint (complex conjugate transpose) of this matrix.
= \$	Clear	Clears this matrix, setting it to a 0x0 matrix.
-= \$	Сору	Makes a new copy, independent copy of this matrix.

= ≬ S	CreMat	Creates an array that can be filled in and passed to the array- based CMat constructor.
= 0 S	CreZer	Creates an array filled with zeros that can be filled in and passed to the array-based CMat constructor.
=♥	Div	Scales this matrix by a constant divisor. This matrix is updated in place, rather than a new matrix being created.
= 0	Dump	Dumps this matrix. Note that there is no limit on the size of the result; every row and column entry is included.
≡0	DumpML	Dumps this matrix in Matlab format for debugging.
= ()	DumpNarrow	Dumps this matrix. This method will produce a relatively compact representation of the matrix.
≡0	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)

= Q	FixUnitary	Makes this matrix closer to being unitary by applying an iterative correction.
- =	GetHashCode	Serves as the default hash function. (Inherited from Object.)
≣♥	GetType	Gets the Type of the current instance. (Inherited from Object.)
≡♥	Kron(CMat)	Computes the Kronecker Product of this matrix with another matrix.
=♥	Kron(Int32, FSharpOptionBoolean)	Computes the Kronecker Product of this matrix with an identity matrix
-= Q	Mul(Double)	Scales this matrix by a constant multiplier. This matrix is updated in place, rather than a new matrix being created.
. ≡ `	Mul(CMat)	Computes the product of this matrix with another matrix. This matrix is on the left-hand side of the product, and the argument matrix is on the right-hand side. This matrix must have the same number of columns as the other matrix has

		rows.
≓ `	Mul(CVec)	Calculates the product of this matrix and a vector. The vector's length must match the numder of columns in this matrix.
≕ ©	Power	Computes an integer power of this matrix. NOTE: This destroys the contents of the current matrix.
≓ ≬ S	Read	Reads a matrix from a stream.
= ♥	ToString	Gets a string representation of this matrix. Note that there is no limit on the size of the result; every row and column entry is included. (Overrides ObjectToString.)
= ♥	UnitaryError	Performs a rough check to see if this matrix is actually unitary. It calculates the biggest deviation from 1 of the diagonal elements of (this adj)*this. In other words, it calculates the largest deviation of the length of a row, viewed as a complex vector, from 1. The orthogonality

		of different rows is not checked.
- :	Write	Writes this matrix to a stream for serialization.

Тор

▲ Operators

	Name	Description
(<u>/-</u> =+) S	AdditionAssignment	Adds one matrix to another.
≝+3 S	DivisionAssignment	Scales a matrix by a constant divisor. The matrix is updated in place, rather than a new matrix being created.
<u>(∕</u> =+) S	Multiply(CMat, CMat)	Computes the product of two matrices. The first matrix must have the same number of columns as the second matrix has rows.
<u>(}-</u> =+} S	Multiply(CMat, CVec)	Computes the product of a matrix and a vector. The vector's length must match the numder of columns in the matrix.
⊻= =+> S	MultiplyAssignment	Scales a matrix by a constant multiplier. The matrix is updated in place, rather than a new matrix being created.



Computes the Kronecker product of two matrices

Тор



Reference Microsoft.Research.Liquid Namespace [T::Microsoft.Research.Liquid.CSMat]

CMat Constructor

Overload List

	Name	Description
≕ Ø	CMat(Double, Double)	Creates a new matrix from arrays of real and imaginary parts.
=≬	CMat(Int32, FSharpListTupleInt32, Int32, Double, Double)	Creates a square matrx from a sparse list of elements.
= ()	CMat(Int32, FSharpOptionBoolean)	Creates a square identity or zero matrix.

Тор

⊿ See Also

Reference CMat Class Microsoft.Research.Liquid Namespace

CMat Constructor (Double, Double)

Creates a new matrix from arrays of real and imaginary parts.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

rs

Type: SystemDouble The real part of each matrix element.

is

Type: SystemDouble The real part of each matrix element.

⊿ See Also

Reference CMat Class CMat Overload Microsoft.Research.Liquid Namespace

CMat Constructor (Int32, FSharpListTupleInt32, Int32, Double, Double)

Creates a square matrx from a sparse list of elements.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

len

Type: SystemInt32

The row (or column) length of the matrix.

xyris

Type: **Microsoft.FSharp.CollectionsFSharpList**TupleInt32, Int32, Double, Double

A list of element location and value tuples, with elements in the order (row,col,real,imag).

⊿ See Also

Reference CMat Class CMat Overload Microsoft.Research.Liquid Namespace

CMat Constructor (Int32, FSharpOptionBoolean)

Creates a square identity or zero matrix.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

n

```
Type: SystemInt32
```

The row (or column) length of the matrix.

zero

Type: Microsoft.FSharp.CoreFSharpOptionBoolean

Option to initialize to a zero matrix rather than to the identity matrix. The default is to initialize to the identity.

⊿ See Also

Reference CMat Class CMat Overload Microsoft.Research.Liquid Namespace

CMat Properties

The CMat type exposes the following members.

▲ Properties

4

	Name	Description
*	Item	Gets an individual element of the matrix, as a Complex number.
*	Length	The number of rows or columns in the matrix, for square matrices.
*	LengthC	The number of columns in the matrix.
	LengthR	The number of rows in the matrix.
Тор		
See Als	0	
Reference		

CMat Class Microsoft.Research.Liquid Namespace

CMatItem Property

Gets an individual element of the matrix, as a Complex number.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



CMatLength Property

The number of rows or columns in the matrix, for square matrices.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____ Copy _ member Length : int with get Property Value Type: Int32

⊿ See Also

Reference CMat Class Microsoft.Research.Liquid Namespace

CMatLengthC Property

The number of columns in the matrix.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

_ Copy _

member LengthC : int with get

Property Value Type: Int32

⊿ See Also

Reference CMat Class Microsoft.Research.Liquid Namespace

CMatLengthR Property

The number of rows in the matrix.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

_ Copy _

member LengthR : int with get

Property Value Type: Int32

⊿ See Also

Reference CMat Class Microsoft.Research.Liquid Namespace

CMat Methods

The CMat type exposes the following members.

Methods

	Name	Description
-=	Add	Adds another matrix to this matrix. The two matrices must have the same number of rows and columns.
-= Q	Adj	Computes the adjoint (complex conjugate transpose) of this matrix.
≓ \$	Clear	Clears this matrix, setting it to a 0x0 matrix.
- =	Сору	Makes a new copy, independent copy of this matrix.
= \$ S	CreMat	Creates an array that can be filled in and passed to the array- based CMat constructor.
≡0 S	CreZer	Creates an array filled with zeros that can be filled in and passed to the array-based CMat constructor.

=∳	Div	Scales this matrix by a constant divisor. This matrix is updated in place, rather than a new matrix being created.
=∳	Dump	Dumps this matrix. Note that there is no limit on the size of the result; every row and column entry is included.
≓©	DumpML	Dumps this matrix in Matlab format for debugging.
≕ ©	DumpNarrow	Dumps this matrix. This method will produce a relatively compact representation of the matrix.
⊒ ∲	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
≝ ©	FixUnitary	Makes this matrix closer to being unitary by applying an iterative correction.
≓∳	GetHashCode	Serves as the default hash function. (Inherited from Object.)
=	GetType	Gets the Type of the

		current instance. (Inherited from Object.)
- = \$	Kron(CMat)	Computes the Kronecker Product of this matrix with another matrix.
. ≓ ©	Kron(Int32, FSharpOptionBoolean)	Computes the Kronecker Product of this matrix with an identity matrix
- = \$	Mul(Double)	Scales this matrix by a constant multiplier. This matrix is updated in place, rather than a new matrix being created.
	Mul(CMat)	Computes the product of this matrix with another matrix. This matrix is on the left-hand side of the product, and the argument matrix is on the right-hand side. This matrix must have the same number of columns as the other matrix has rows.
. = \$	Mul(CVec)	Calculates the product of this matrix and a vector. The vector's length must match the numder of columns in this matrix.
≓∳	Power	Computes an integer power of this matrix. NOTE: This destroys the

		contents of the current matrix.
⊴ 0 S	Read	Reads a matrix from a stream.
≡	ToString	Gets a string representation of this matrix. Note that there is no limit on the size of the result; every row and column entry is included (Overrides ObjectToString.)
	UnitaryError	Performs a rough check to see if this matrix is actually unitary. It calculates the biggest deviation from 1 of the diagonal elements of (this adj)*this. In other words, it calculates the largest deviation of the length of a row, viewed as a complex vector, from 1. The orthogonality of different rows is not checked.
		Writes this matrix to a
Reference CMat Class Microsoft.Research.Liquid Namespace

CMatAdd Method

Adds another matrix to this matrix. The two matrices must have the same number of rows and columns.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax F# member Add : m2 : CMat -> unit

Parameters

m2

Type: Microsoft.Research.LiquidCMat The matrix to add to this one.

⊿ See Also

Reference CMat Class Microsoft.Research.Liquid Namespace

CMatAdj Method

Computes the adjoint (complex conjugate transpose) of this matrix.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____ Copy ____ member Adj : unit -> CMat

Return Value Type: CMat The adjoint matrix

⊿ See Also

Reference CMat Class Microsoft.Research.Liquid Namespace

CMatClear Method

Clears this matrix, setting it to a 0x0 matrix.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# Copy

member Clear : unit -> unit

⊿ See Also

Reference CMat Class Microsoft.Research.Liquid Namespace

CMatCopy Method

Makes a new copy, independent copy of this matrix.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

member Copy : unit -> CMat

Return Value Type: CMat The new matrix

⊿ See Also

Reference CMat Class Microsoft.Research.Liquid Namespace

CMatCreMat Method

Creates an array that can be filled in and passed to the array-based CMat constructor.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax F# Copy static member CreMat : lenR : int * lenC : int * f : FSharpFunc<int, FSharpFunc<int, float</pre> 4 Þ **Parameters** lenR Type: SystemInt32 The number of rows lenC Type: SystemInt32 The number of columns f Type: Microsoft.FSharp.CoreFSharpFuncInt32, FSharpFuncInt32, Double The function to call to compute the elements of the array. It will get called with the row first and then the column (curried).

Return Value Type: Double

The new array

⊿ See Also

Reference

CMat Class Microsoft.Research.Liquid Namespace

CMatCreZer Method

Creates an array filled with zeros that can be filled in and passed to the array-based CMat constructor.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F#
        Copy
static member CreZer :
        lenR : int *
        lenC : int -> float[][]
```

Parameters

lenR

Type: SystemInt32 The number of rows *lenC* Type: SystemInt32 The number of columns

Return Value Type: Double The new array

⊿ See Also

Reference CMat Class Microsoft.Research.Liquid Namespace

CMatDiv Method

Scales this matrix by a constant divisor. This matrix is updated in place, rather than a new matrix being created.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# _____Copy _____
member Div :
        div : float -> CMat
```

Parameters

div

Type: SystemDouble The scale divisor. The matrix is multiplied by 1/div.

Return Value Type: CMat This matrix, as updated after scaling

▲ See Also

Reference CMat Class Microsoft.Research.Liquid Namespace

CMatDump Method

Dumps this matrix. Note that there is no limit on the size of the result; every row and column entry is included.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#	у _
<pre>member Dump : f : FSharpOption<fsharpfunc<int, :="" fsharp="" fsharpoption<int="" level=""> -> unit</fsharpfunc<int,></pre>	Fι
<u> ۱</u>	▶

Parameters

f

Type: Microsoft.FSharp.CoreFSharpOptionFSharpFuncInt32, FSharpFuncString, Unit

The optional output function to use. The default is showLogInd. *level*

Type: Microsoft.FSharp.CoreFSharpOptionInt32

The optional indentation level. The default is 0.

⊿ See Also

Reference

CMat Class Microsoft.Research.Liquid Namespace

CMatDumpML Method

Dumps this matrix in Matlab format for debugging.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

f

Type: Microsoft.FSharp.CoreFSharpOptionFSharpFuncInt32, FSharpFuncString, Unit

The optional output function to use. The default is showLogInd. *level*

Type: Microsoft.FSharp.CoreFSharpOptionInt32

The optional indentation level. The default is 0.

prec

Type: Microsoft.FSharp.CoreFSharpOptionInt32 precision 0=low 1=normal 2=high 3=full (optional=1)

⊿ See Also

Reference

CMat Class Microsoft.Research.Liquid Namespace

CMatDumpNarrow Method

Dumps this matrix. This method will produce a relatively compact representation of the matrix.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

f

Type: Microsoft.FSharp.CoreFSharpOptionFSharpFuncInt32, FSharpFuncString, Unit

The optional output function to use. The default is showLogInd. *level*

Type: Microsoft.FSharp.CoreFSharpOptionInt32

The optional indentation level. The default is 0.

⊿ See Also

Reference

CMat Class Microsoft.Research.Liquid Namespace

CMatFixUnitary Method

Makes this matrix closer to being unitary by applying an iterative correction.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

tol

Type: Microsoft.FSharp.CoreFSharpOptionDouble

An optional desired tolerance for the deviation from unitarity, as measured by UnitaryError. The default is 1.0e-13.

maxIter

Type: Microsoft.FSharp.CoreFSharpOptionInt32

An optional maximum number of iterations to perform. The default is 5.

Return Value

Type: TupleInt32, Double

A tuple of the number of iterations performed, for cost estimates, and the final deviation from unitarity.

⊿ See Also

Reference CMat Class Microsoft.Research.Liquid Namespace

CMatKron Method

Overload List

	Name	Description
≡©	Kron(CMat)	Computes the Kronecker Product of this matrix with another matrix.
≓ ©	Kron(Int32, FSharpOptionBoolean)	Computes the Kronecker Product of this matrix with an identity matrix
Тор		

⊿ See Also

Reference CMat Class Microsoft.Research.Liquid Namespace

CMatKron Method (CMat)

Computes the Kronecker Product of this matrix with another matrix.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

m2

Type: Microsoft.Research.LiquidCMat The right-hand side matrix in the Kronecker product.

Return Value Type: CMat The result of the Kronecker product

⊿ See Also

Reference CMat Class Kron Overload Microsoft.Research.Liquid Namespace

CMatKron Method (Int32, FSharpOptionBoolean)

Computes the Kronecker Product of this matrix with an identity matrix

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

len

Type: SystemInt32

The dimension of the identity matrix (N of NxN).

left

Type: Microsoft.FSharp.CoreFSharpOptionBoolean

An option that, if true, specifies that the identity matrix should be on the left side of the Kronecker product. The default is false, which means that the identity is on the right.

Return Value

Type: CMat The result of the Kronecker product

⊿ See Also

Reference

CMat Class Kron Overload Microsoft.Research.Liquid Namespace

CMatMul Method

Overload List

	Name	Description
.≡∳	Mul(Double)	Scales this matrix by a constant multiplier. This matrix is updated in place, rather than a new matrix being created.
≕	Mul(CMat)	Computes the product of this matrix with another matrix. This matrix is on the left-hand side of the product, and the argument matrix is on the right-hand side. This matrix must have the same number of columns as the other matrix has rows.
≡	Mul(CVec)	Calculates the product of this matrix and a vector. The vector's length must match the numder of columns in this matrix.

Тор

⊿ See Also

Reference CMat Class Microsoft.Research.Liquid Namespace

CMatMul Method (Double)

Scales this matrix by a constant multiplier. This matrix is updated in place, rather than a new matrix being created.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

mul

Type: SystemDouble The scale factor to multiply by.

Return Value Type: CMat This matrix, as updated after scaling

▲ See Also

Reference CMat Class Mul Overload Microsoft.Research.Liquid Namespace

CMatMul Method (CMat)

Computes the product of this matrix with another matrix. This matrix is on the left-hand side of the product, and the argument matrix is on the right-hand side. This matrix must have the same number of columns as the other matrix has rows.

Copy

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F#
```

member Mul :
 m2 : CMat -> CMat

Parameters

m2

Type: Microsoft.Research.LiquidCMat The second matrix.

Return Value Type: CMat The resulting product matrix.

⊿ See Also

Reference CMat Class Mul Overload Microsoft.Research.Liquid Namespace

CMatMul Method (CVec)

Calculates the product of this matrix and a vector. The vector's length must match the number of columns in this matrix.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

V

Type: Microsoft.Research.LiquidCVec The vector to multiply

Return Value Type: CVec The resulting product vector

J See Also

Reference CMat Class Mul Overload Microsoft.Research.Liquid Namespace

CMatPower Method

Computes an integer power of this matrix. NOTE: This destroys the contents of the current matrix.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

n

Type: SystemInt32 The power to raise this matrix to; must be greater than 0.

Return Value Type: CMat The resulting matrix

J See Also

Reference CMat Class Microsoft.Research.Liquid Namespace

CMatRead Method

Reads a matrix from a stream.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

br

Type: System.IOBinaryReader A BinaryReader to deserialize a matrix from.

Return Value Type: CMat The new mnatrix

⊿ See Also

Reference CMat Class Microsoft.Research.Liquid Namespace

CMatToString Method

Gets a string representation of this matrix. Note that there is no limit on the size of the result; every row and column entry is included.

Copy

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

abstract ToString : unit -> string
override ToString : unit -> string

Return Value Type: String The string representation

⊿ See Also

Reference CMat Class Microsoft.Research.Liquid Namespace

CMatUnitaryError Method

Performs a rough check to see if this matrix is actually unitary. It calculates the biggest deviation from 1 of the diagonal elements of (this adj)*this. In other words, it calculates the largest deviation of the length of a row, viewed as a complex vector, from 1. The orthogonality of different rows is not checked.

Namespace: Microsoft.Research.Liquid

Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

member UnitaryError : unit -> float

Return Value Type: Double The worst deviation from 1.

⊿ See Also

Reference CMat Class Microsoft.Research.Liquid Namespace

CMatWrite Method

Writes this matrix to a stream for serialization.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



bw

Type: System.IOBinaryWriter A BinaryWriter to serialize this matrix to.

⊿ See Also

Reference CMat Class Microsoft.Research.Liquid Namespace

CMat Operators

The CMat type exposes the following members.

▲ Operators

	Name	Description
=+> S	AdditionAssignment	Adds one matrix to another.
(<u>~-</u> =+) S	DivisionAssignment	Scales a matrix by a constant divisor. The matrix is updated in place, rather than a new matrix being created.
<u>(∕</u> =+) S	Multiply(CMat, CMat)	Computes the product of two matrices. The first matrix must have the same number of columns as the second matrix has rows.
(<u>/-</u> =+) S	Multiply(CMat, CVec)	Computes the product of a matrix and a vector. The vector's length must match the numder of columns in the matrix.
<u>(/-</u> =+) S	MultiplyAssignment	Scales a matrix by a constant multiplier. The matrix is updated in place, rather than a new matrix being created.
(<u>/-</u> =+) S	MultiplyBang	Computes the Kronecker

Тор

⊿ See Also

Reference CMat Class Microsoft.Research.Liquid Namespace

CMatAdditionAssignment Operator

Adds one matrix to another.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

F# does not support this operator.

Parameters

m1

Type: Microsoft.Research.LiquidCMat The left matrix. This matrix is modified.

m2

Type: Microsoft.Research.LiquidCMat The right matrix

⊿ See Also

Reference CMat Class Microsoft.Research.Liquid Namespace

CMatDivisionAssignment Operator

Scales a matrix by a constant divisor. The matrix is updated in place, rather than a new matrix being created.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

F# does not support this operator.

Parameters

m1

Type: Microsoft.Research.LiquidCMat The matrix to scale.

div

Type: SystemDouble The scale divisor. The matrix is multiplied by 1/div.

⊿ See Also

Reference CMat Class Microsoft.Research.Liquid Namespace

CMatMultiply Operator

Overload List

	Name	Description
≝+) S	Multiply(CMat, CMat)	Computes the product of two matrices. The first matrix must have the same number of columns as the second matrix has rows.
(<u>∧</u> = =+) S	Multiply(CMat, CVec)	Computes the product of a matrix and a vector. The vector's length must match the numder of columns in the matrix.
Тор		

⊿ See Also

Reference CMat Class Microsoft.Research.Liquid Namespace

CMatMultiply Operator (CMat, CMat)

Computes the product of two matrices. The first matrix must have the same number of columns as the second matrix has rows.

Copy

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

```
static let inline (*)
    m1 : CMat *
    m2 : CMat : CMat
```

Parameters

m1

Type: Microsoft.Research.LiquidCMat The left (first) matrix

*m*2

Type: Microsoft.Research.LiquidCMat The right (second) matrix

Return Value Type: CMat The resulting product matrix

⊿ See Also

Reference CMat Class
Multiply Overload Microsoft.Research.Liquid Namespace

CMatMultiply Operator (CMat, CVec)

Computes the product of a matrix and a vector. The vector's length must match the number of columns in the matrix.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

```
static let inline (*)
    m : CMat *
    v : CVec : CVec
```

Parameters

т

Type: Microsoft.Research.LiquidCMat The matrix

V

Type: Microsoft.Research.LiquidCVec The vector

Return Value Type: CVec The resulting product vector

⊿ See Also

Reference CMat Class Multiply Overload Microsoft.Research.Liquid Namespace

CMatMultiplyAssignment Operator

Scales a matrix by a constant multiplier. The matrix is updated in place, rather than a new matrix being created.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

F# does not support this operator.

Parameters

m1

Type: Microsoft.Research.LiquidCMat The matrix to scale.

mul

Type: SystemDouble The constant to multiply by.

J See Also

Reference CMat Class Microsoft.Research.Liquid Namespace

CMatMultiplyBang Operator

Computes the Kronecker product of two matrices

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

_ Сору _

F# does not support this operator.

Parameters

m1

Type: Microsoft.Research.LiquidCMat The left matrix

m2

Type: Microsoft.Research.LiquidCMat The right matrix

Return Value Type: CMat The result of the Kronecker product

⊿ See Also

Reference CMat Class Microsoft.Research.Liquid Namespace

Complex Structure

Data type for complex numbers.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#	Сору
[<sealedattribute>] [<serializableattribute>] type Complex =</serializableattribute></sealedattribute>	
struct	
<pre>interface IEquatable<complex></complex></pre>	
<pre>interface IStructuralEquatable</pre>	
<pre>interface IComparable<complex></complex></pre>	
interface IComparable	
<pre>interface IStructuralComparable</pre>	
end	

The Complex type exposes the following members.

Constructors

	Name	Description
= 0	Complex	Constructs a complex number from its real and imaginary parts.

Тор

Properties

	Name	Description
*	i	Gets the imaginary part of this Complex number.
🖀 S	I	The square root of negative one, as a Complex number.
*	MCC	Gets the squared magnitude of this Complex number.
🖹 S	One	One, as a Complex number
*	r	Gets the real part of this Complex number.
🖀 s	Tol	Tolerance for comparing two Complex numbers. This is used in various places in the system.
🖻 s	Zero	Zero, as a Complex number

Тор

▲ Methods

	Name	Description
= 0	Conj	Computes the complex conjugate of this Complex number.
=♥	Сору	Duplicates this Complex number in a new instance.
≓∳	Dump	Dumps this Complex number to the console and log with an optional indentation.
≡Ŵ	Equals	Indicates whether this instance

		and a specified object are equal. (Inherited from ValueType.)
-= Q	GetHashCode	Returns the hash code for this instance. (Inherited from ValueType.)
-= Q	GetType	Gets the Type of the current instance. (Inherited from Object.)
-= 🏟	Narrow	Converts this Complex number to a short human-readable string.
-= Q	ToString	Converts this Complex number to a human-readable string. (Overrides ValueTypeToString.)
- ≓∳	Wide	Converts this Complex number to a string suitable for input to other programs.

Тор

▲ Operators

	Name	Description
(/ =+) S	Addition	Adds two Complex numbers.
≝+) S	Multiply(Double, Complex)	Multiplies a real and a Complex number.
<u>∞–</u> =+) S	Multiply(Complex, Complex)	Multiplies two Complex numbers.
<u>(∕-</u> =+) S	Multiply(Complex, Double)	Multiplies a Complex and a real number.

en s	Subtraction	Subtracts one Complex number from another.
≝⇔ S	TwiddleTwiddle	Gets the complex conjugate of a Complex number.
Kan S	UnaryNegation	Gets the negation of a Complex number.
Тор		
See A	lso	

Reference Microsoft.Research.Liquid Namespace

Complex Constructor

Constructs a complex number from its real and imaginary parts.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Complex Properties

The Complex type exposes the following members.

▲ Properties

	Name	Description
	i	Gets the imaginary part of this Complex number.
≌ S	I	The square root of negative one, as a Complex number.
*	MCC	Gets the squared magnitude of this Complex number.
🖀 S	One	One, as a Complex number
*	r	Gets the real part of this Complex number.
🖹 S	Tol	Tolerance for comparing two Complex numbers. This is used in various places in the system.
🖹 S	Zero	Zero, as a Complex number

Тор

⊿ See Also

Reference Complex Structure Microsoft.Research.Liquid Namespace

Complexi Property

Gets the imaginary part of this Complex number.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

member i : float with get

Property Value Type: Double The imaginary part

⊿ See Also

Reference Complex Structure Microsoft.Research.Liquid Namespace

ComplexI Property

The square root of negative one, as a Complex number.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

 F#
 Copy

 static member I : Complex with get

 Property Value

 Type: Complex

 A See Also

Reference Complex Structure Microsoft.Research.Liquid Namespace

ComplexMCC Property

Gets the squared magnitude of this Complex number.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

member MCC : float with get

Return Value Type: Double The squared magnitude (r*r+i*i)

▲ See Also

Reference Complex Structure Microsoft.Research.Liquid Namespace

ComplexOne Property

One, as a Complex number

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Microsoft.Research.Liquid Namespace

Complexr Property

Gets the real part of this Complex number.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

___ Сору __

member r : float with get

Property Value Type: Double The real part

⊿ See Also

Reference Complex Structure Microsoft.Research.Liquid Namespace

ComplexTol Property

Tolerance for comparing two Complex numbers. This is used in various places in the system.

_ Copy _

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

✓ Syntax

static member Tol : float with get, set

Property Value Type: Double

▲ See Also

Reference Complex Structure Microsoft.Research.Liquid Namespace

ComplexZero Property

Zero, as a Complex number

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору _

static member Zero : Complex with get

Property Value Type: Complex

⊿ See Also

Reference Complex Structure Microsoft.Research.Liquid Namespace

Complex Methods

The Complex type exposes the following members.

Methods

	Name	Description
=∳	Conj	Computes the complex conjugate of this Complex number.
=♥	Сору	Duplicates this Complex number in a new instance.
=0	Dump	Dumps this Complex number to the console and log with an optional indentation.
≓ ©	Equals	Indicates whether this instance and a specified object are equal. (Inherited from ValueType.)
≓ \$	GetHashCode	Returns the hash code for this instance. (Inherited from ValueType.)
≝\$	GetType	Gets the Type of the current instance. (Inherited from Object.)
≡∳	Narrow	Converts this Complex number to a short human-readable string.
⊒ ≬	ToString	Converts this Complex number to a human-readable string.

		(Overrides ValueTypeToString.)
≓Ŵ	Wide	Converts this Complex number to a string suitable for input to other programs.
Тор		
⊿ See /	Also	
Referen Complex Microsof	nce < Structure ft.Research.Liqu	d Namespace

ComplexConj Method

Computes the complex conjugate of this Complex number.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy _

▲ Syntax

F#

member Conj : unit -> Complex

Return Value Type: Complex The conjugate

⊿ See Also

Reference Complex Structure Microsoft.Research.Liquid Namespace

ComplexCopy Method

Duplicates this Complex number in a new instance.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Copy _

member Copy : unit -> Complex

Return Value Type: Complex The new Complex

⊿ See Also

Reference Complex Structure Microsoft.Research.Liquid Namespace

ComplexDump Method

Dumps this Complex number to the console and log with an optional indentation.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

f

Type: Microsoft.FSharp.CoreFSharpOptionFSharpFuncInt32, FSharpFuncString, Unit

The optional output function to use. The default is showLogInd. *level*

Type: Microsoft.FSharp.CoreFSharpOptionInt32

The optional indentation level. The default is 0.

⊿ See Also

Reference Complex Structure Microsoft.Research.Liquid Namespace

ComplexNarrow Method

Converts this Complex number to a short human-readable string.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

_ Copy _

member Narrow : unit -> string

Return Value Type: String The formatted string representation

⊿ See Also

Reference Complex Structure Microsoft.Research.Liquid Namespace

ComplexToString Method

Converts this Complex number to a human-readable string.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Copy

```
abstract ToString : unit -> string
override ToString : unit -> string
```

Return Value Type: String The formatted string representation

⊿ See Also

Reference Complex Structure Microsoft.Research.Liquid Namespace

ComplexWide Method

Converts this Complex number to a string suitable for input to other programs.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

____ Сору

▲ Syntax

F#

member Wide : unit -> string

Return Value Type: String The formatted string representation

⊿ See Also

Reference Complex Structure Microsoft.Research.Liquid Namespace

Complex Operators

The Complex type exposes the following members.

▲ Operators

	Name	Description
=+> S	Addition	Adds two Complex numbers.
<u>⊱</u> =+> S	Multiply(Double, Complex)	Multiplies a real and a Complex number.
<u>\/</u> = =+} S	Multiply(Complex, Complex)	Multiplies two Complex numbers.
<u>\∕-</u> =+ S	Multiply(Complex, Double)	Multiplies a Complex and a real number.
<u>\/</u> = =+} S	Subtraction	Subtracts one Complex number from another.
≝÷ S	TwiddleTwiddle	Gets the complex conjugate of a Complex number.
K≓ ≓⇔ S	UnaryNegation	Gets the negation of a Complex number.

Тор

⊿ See Also

Reference Complex Structure Microsoft.Research.Liquid Namespace

ComplexAddition Operator

Adds two Complex numbers.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

а

Type: Microsoft.Research.LiquidComplex The left-hand operand

b

Type: Microsoft.Research.LiquidComplex The right-hand operand

Return Value

Type: Complex NThe sum of the two operands

⊿ See Also

Reference Complex Structure Microsoft.Research.Liquid Namespace

ComplexMultiply Operator

Overload List

	Name	Description
<u>(≻-</u> =+) S	Multiply(Double, Complex)	Multiplies a real and a Complex number.
≝⇔ S	Multiply(Complex, Complex)	Multiplies two Complex numbers.
≝÷ S	Multiply(Complex, Double)	Multiplies a Complex and a real number.

Тор

⊿ See Also

Reference Complex Structure Microsoft.Research.Liquid Namespace

ComplexMultiply Operator (Double, Complex)

Multiplies a real and a Complex number.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

Parameters

а

Type: SystemDouble The real number

b

Type: Microsoft.Research.LiquidComplex The Complex number

Return Value

Type: Complex The product of the two numbers

▲ See Also

Reference Complex Structure Multiply Overload Microsoft.Research.Liquid Namespace

ComplexMultiply Operator (Complex, Complex)

Multiplies two Complex numbers.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

а

Type: Microsoft.Research.LiquidComplex Left hand operand.

b

Type: Microsoft.Research.LiquidComplex Right hand operand.

Return Value

Type: Complex The product of the two numbers

▲ See Also

Reference Complex Structure Multiply Overload Microsoft.Research.Liquid Namespace
ComplexMultiply Operator (Complex, Double)

Multiplies a Complex and a real number.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F#

static let inline (*)

a : Complex *

b : float : Complex
```

Copy

Parameters

а

Type: Microsoft.Research.LiquidComplex The Complex number

b

Type: SystemDouble The real number

Return Value

Type: Complex The product of the two numbers

▲ See Also

Reference Complex Structure Multiply Overload Microsoft.Research.Liquid Namespace

ComplexSubtraction Operator

Subtracts one Complex number from another.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

а

Type: Microsoft.Research.LiquidComplex The left-hand operand

b

Type: Microsoft.Research.LiquidComplex The right-hand operand

Return Value

Type: Complex The difference of the two operands

⊿ See Also

Reference Complex Structure Microsoft.Research.Liquid Namespace

ComplexTwiddleTwiddle Operator

Gets the complex conjugate of a Complex number.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

F# does not support this operator.

Parameters

а

Type: Microsoft.Research.LiquidComplex The Complex number to conjugate.

Return Value

Type: Complex The conjugated Complex number

▲ See Also

Reference Complex Structure Microsoft.Research.Liquid Namespace

ComplexUnaryNegation Operator

Gets the negation of a Complex number.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____ Copy static let inline (-) a : Complex : Complex

Parameters

а

Type: Microsoft.Research.LiquidComplex The Complex number to negate

Return Value Type: Complex The negated Complex number, (-r,-i)

⊿ See Also

Reference Complex Structure Microsoft.Research.Liquid Namespace

CSMat Class

A sparse matrix of complex numbers.

▲ Inheritance Hierarchy

SystemObject Microsoft.Research.LiquidCSMat

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

```
[<SerializableAttribute>]
type CSMat = class end
```

The CSMat type exposes the following members.

Constructors

	Name	Description
≓ ≬	CSMat(Int32, FSharpListTupleInt32, Int32, Double, Double)	Creates a matrix from a list of elements.
≡ ©	CSMat(Int32, FSharpOptionBoolean)	Creates a new square identity or zero matrix.
≡ ©	CSMat(CMat, FSharpOptionDouble)	Creates a sparse matrix from a dense matrix.

Тор

▲ Properties

	Name	Description
	Item	Gets an element of the matrix.
*	Length	The dimension of this matrix; that is, the number of rows or columns.

Тор

▲ Methods

	Name	Description
= \$	Adj	Computes the adjoint (complex conjugate transpose) of this matrix.
=♥	Clear	Clears out this matrix, setting all elements to zero.
≓ ≬	Сору	Makes a copy of this matrix.
≝∳	Dense	Creates a dense matrix from this sparse matrix.
=♥	Dump	Dumps this matrix as a list of row-column indices and element complex values. Note that all non-zero elements are listed, so the output may be quite long.

	DumpDense	Dumps this sparse matrix in the same format as a dense matrix.
=♥	DumpMCC	Dumps this matrix as a list of row-column indices and element squared magnitudes. Note that all non-zero elements are listed, so the output may be quite long.
. ≓ ©	DumpML	Dumps this matrix in MatLab format.
=♥	Equals(Object)	Determines whether the specified object is equal to the current object. (Inherited from Object.)
. ≓ ©	Equals(CSMat, FSharpOptionDouble)	Determines whether this matrix is equal to another.
. ≓ ∲	Filled	Return indicies of filled entries in the sparse matrix (may include zeros)
. ≕	GetHashCode	Serves as the default hash function. (Inherited from Object.)
. ≓ ©	GetType	Gets the Type of the current instance. (Inherited from Object.)
=♥	Kron(Int32)	Computes the Kronecker Product of this matrix with an identity matrix. The

		identity matrix is on the right-hand side of the product.
≓Q	Kron(CSMat)	Computes the Kronecker Product of this matrix with another matrix.
≓ ∳	Mul(CSMat)	Computes the product of this matrix and another matrix. Highly efficient.
≡ ≬	Mul(CVec)	Computes the product of this matric and a vector.
⊧ \$ S	Read	Reads a matrix from a stream. The matrix must originally have been written using the Write method.
. ≓	ToString	Gets a string representation of this matrix. Note that all non- zero elements are listed, so this string may be quite long. (Overrides ObjectToString.)
≟∲	UnitaryError	Performs a rough check to see if this matrix is actually unitary. It calculates the biggest deviation from 1 of the diagonal elements of (this adj)*this. In other words, it calculates the largest

		deviation of the length of a row, viewed as a complex vector, from 1. The orthogonality of different rows is not checked.
-=	Write	Writes this matrix to a stream. The matrix may be recreated by using the Read method.

Тор

▲ Operators

	Name	Description
(<u>/-</u> =+) S	Multiply(CSMat, CSMat)	Computes the product of two matrices. Highly efficient.
(/- =+) S	Multiply(CSMat, CVec)	Computes the product of a matrix and a vector.
(<u>/-</u> =+) S	MultiplyBang	Computes the Kronecker product of two matrices

Тор

⊿ See Also

Reference Microsoft.Research.Liquid Namespace [T::Microsoft.Research.Liquid.CMat]

CSMat Constructor

Overload List

	Name	Description
. ≕	CSMat(Int32, FSharpListTupleInt32, Int32, Double, Double)	Creates a matrix from a list of elements.
≓∲	CSMat(Int32, FSharpOptionBoolean)	Creates a new square identity or zero matrix.
÷	CSMat(CMat, FSharpOptionDouble)	Creates a sparse matrix from a dense matrix.

Тор

⊿ See Also

Reference CSMat Class Microsoft.Research.Liquid Namespace

CSMat Constructor (Int32, FSharpListTupleInt32, Int32, Double, Double)

Creates a matrix from a list of elements.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

len

Type: SystemInt32

The dimension of the matrix; that is, the row or column count. *xyris*

Type: **Microsoft.FSharp.CollectionsFSharpList**TupleInt32, Int32, Double, Double

A list of elements. Each element should be a tuple in the form (row,col,real,imag).

⊿ See Also

Reference CSMat Class CSMat Overload Microsoft.Research.Liquid Namespace

CSMat Constructor (Int32, FSharpOptionBoolean)

Creates a new square identity or zero matrix.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



zero : FSharpOption<bool> -> CSMat

Parameters

п

Type: SystemInt32

The dimension of the matrix; that is, the row or column count.

zero

Type: Microsoft.FSharp.CoreFSharpOptionBoolean

An option to create a zero matrix rather than an identity matrix, if true. The default is false, which creates an identity matrix.

⊿ See Also

```
Reference
CSMat Class
CSMat Overload
Microsoft.Research.Liquid Namespace
```

CSMat Constructor (CMat, FSharpOptionDouble)

Creates a sparse matrix from a dense matrix.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

т

Type: Microsoft.Research.LiquidCMat The source matrix.

tol

Type: **Microsoft.FSharp.CoreFSharpOption**Double An optional tolerance for identifying zero elements. The default is to use Complex.tol as the maximum magnitude to consider 0.

⊿ See Also

Reference CSMat Class CSMat Overload Microsoft.Research.Liquid Namespace

CSMat Properties

The CSMat type exposes the following members.

▲ Properties

		Name	Description
	1	Item	Gets an element of the matrix.
	7	Length	The dimension of this matrix; that is, the number of rows or columns.
Тс	р		
⊿ S	See Also		
R	eference		

CSMat Class Microsoft.Research.Liquid Namespace

CSMatItem Property

Gets an element of the matrix.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

 F#
 Copy

 member Item : Complex with get, set
 Copy

 Parameters
 X

 X
 Type: SystemInt32 The row index of the desired element.

 Y
 Type: SystemInt32 The column index of the desired element.

Return Value Type: Complex The matrix element, as a Complex number

⊿ See Also

Reference CSMat Class Microsoft.Research.Liquid Namespace

CSMatLength Property

The dimension of this matrix; that is, the number of rows or columns.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____ Copy ____ member Length : int with get _____ Property Value

Type: Int32

⊿ See Also

Reference CSMat Class Microsoft.Research.Liquid Namespace

CSMat Methods

The CSMat type exposes the following members.

Methods

	Name	Description
=0	Adj	Computes the adjoint (complex conjugate transpose) of this matrix.
=♥	Clear	Clears out this matrix, setting all elements to zero.
=♥	Сору	Makes a copy of this matrix.
= Q	Dense	Creates a dense matrix from this sparse matrix.
≓ ∲	Dump	Dumps this matrix as a list of row-column indices and element complex values. Note that all non-zero elements are listed, so the output may be quite long.
≡ 0	DumpDense	Dumps this sparse matrix in the same format as a dense matrix.
÷	DumpMCC	Dumps this matrix as a list of row-column indices and

		element squared magnitudes. Note that all non-zero elements are listed, so the output may be quite long.
= \$	DumpML	Dumps this matrix in MatLab format.
=∲	Equals(Object)	Determines whether the specified object is equal to the current object. (Inherited from Object.)
=0	Equals(CSMat, FSharpOptionDouble)	Determines whether this matrix is equal to another.
= \$	Filled	Return indicies of filled entries in the sparse matrix (may include zeros)
≓ \$	GetHashCode	Serves as the default hash function. (Inherited from Object.)
= Q	GetType	Gets the Type of the current instance. (Inherited from Object.)
≡Ŵ	Kron(Int32)	Computes the Kronecker Product of this matrix with an identity matrix. The identity matrix is on the right-hand side of the product.
= Q	Kron(CSMat)	Computes the Kronecker

		Product of this matrix with another matrix.
=♥	Mul(CSMat)	Computes the product of this matrix and another matrix. Highly efficient.
= 0	Mul(CVec)	Computes the product of this matric and a vector.
≓Ŷ S	Read	Reads a matrix from a stream. The matrix must originally have been written using the Write method.
=♥	ToString	Gets a string representation of this matrix. Note that all non- zero elements are listed, so this string may be quite long. (Overrides ObjectToString.)
= ♥	UnitaryError	Performs a rough check to see if this matrix is actually unitary. It calculates the biggest deviation from 1 of the diagonal elements of (this adj)*this. In other words, it calculates the largest deviation of the length of a row, viewed as a complex vector, from 1. The orthogonality of different rows is not checked.

=0	Write	Writes this matrix to a stream. The matrix may be recreated by using the Read method.
Тор		
See	Also	
Referen CSMat (Microsof	1Ce Class ft.Research.Liquid N	amespace

CSMatAdj Method

Computes the adjoint (complex conjugate transpose) of this matrix.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# ______ Copy _____ member Adj : unit -> CSMat

Return Value Type: CSMat A new matrix that is the adjoint of this matrix

⊿ See Also

Reference CSMat Class Microsoft.Research.Liquid Namespace

CSMatClear Method

Clears out this matrix, setting all elements to zero.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



⊿ See Also

Reference CSMat Class Microsoft.Research.Liquid Namespace

CSMatCopy Method

Makes a copy of this matrix.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

_ Сору _

member Copy : unit -> CSMat

Return Value Type: CSMat The new matrix

⊿ See Also

Reference CSMat Class Microsoft.Research.Liquid Namespace

CSMatDense Method

Creates a dense matrix from this sparse matrix.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

zeros

Type: Microsoft.FSharp.CoreFSharpOptionBoolean

Option to create a dense matrix the same size as this matrix but filled with zeroes, rather than copying this matrix. The default is to copy the entries of this matrix to the new dense matrix.

Return Value Type: CMat The new dense matrix

⊿ See Also

Reference CSMat Class Microsoft.Research.Liquid Namespace

CSMatDump Method

Dumps this matrix as a list of row-column indices and element complex values. Note that all non-zero elements are listed, so the output may be quite long.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

f

Type: Microsoft.FSharp.CoreFSharpOptionFSharpFuncInt32, FSharpFuncString, Unit

The optional output function to use. The default is showLogInd. *level*

Type: **Microsoft.FSharp.CoreFSharpOptionInt32** The optional indentation level. The default is 0.

⊿ See Also

Reference

CSMat Class Microsoft.Research.Liquid Namespace

CSMatDumpDense Method

Dumps this sparse matrix in the same format as a dense matrix.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# Copy member DumpDense : f : FSharpOption<FSharpFunc<int, FSharpFu</pre> level : FSharpOption<int> -> unit 4 F

Parameters

f

Type: Microsoft.FSharp.CoreFSharpOptionFSharpFuncInt32, FSharpFuncString, Unit

The optional output function to use. The default is showLogInd. *level*

Type: **Microsoft.FSharp.CoreFSharpOptionInt32** The optional indentation level. The default is 0.

⊿ See Also

Reference

CSMat Class Microsoft.Research.Liquid Namespace

CSMatDumpMCC Method

Dumps this matrix as a list of row-column indices and element squared magnitudes. Note that all non-zero elements are listed, so the output may be quite long.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# _____Copy _____Copy _____Copy _____Copy _____Copy ______Copy _____COPY ______COPY _____COPY ______COPY _____COPY ______COPY _____COPY ______COPY ______COPY ______COPY _____COPY ______COPY ______COPY ______COPY ______COPY ______COPY _____COPY ____COPY _____COPY _____COPY ____COPY ___COPY ___COPY ____COPY __
```

Parameters

f

Type: Microsoft.FSharp.CoreFSharpOptionFSharpFuncInt32, FSharpFuncString, Unit

The optional output function to use. The default is showLogInd. *level*

Type: **Microsoft.FSharp.CoreFSharpOptionInt32** The optional indentation level. The default is 0.

⊿ See Also

Reference

CSMat Class Microsoft.Research.Liquid Namespace

CSMatDumpML Method

Dumps this matrix in MatLab format.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

f

Type: Microsoft.FSharp.CoreFSharpOptionFSharpFuncInt32, FSharpFuncString, Unit

The optional output function to use. The default is showLogInd. *level*

Type: Microsoft.FSharp.CoreFSharpOptionInt32

The optional indentation level. The default is 0.

nam

Type: Microsoft.FSharp.CoreFSharpOptionString

An optional name for the matrix. The default is A. *idxOffset*

Type: Microsoft.FSharp.CoreFSharpOptionInt32

An optional number of rows and columns to skip. If this is not zero, then the square submatrix starting at this offset is dumped instead of the full matrix. The default is 0.

⊿ See Also

Reference

CSMat Class Microsoft.Research.Liquid Namespace
CSMatEquals Method

Overload List

	Name	Description
≓ ©	Equals(Object)	Determines whether the specified object is equal to the current object. (Inherited from Object.)
≓ ∲	Equals(CSMat, FSharpOptionDouble)	Determines whether this matrix is equal to another.
Тор		

⊿ See Also

Reference CSMat Class Microsoft.Research.Liquid Namespace

CSMatEquals Method (CSMat, FSharpOptionDouble)

Determines whether this matrix is equal to another.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# _____Copy ____
member Equals :
    m2 : CSMat *
    tol : FSharpOption<float> -> bool
```

Parameters

m2

Type: Microsoft.Research.LiquidCSMat

The sparse matrix to compare to.

tol

Type: Microsoft.FSharp.CoreFSharpOptionDouble

An optional tolerance for considering real or imaginary parts equal. The default is Complex.Tol.

Return Value

Type: Boolean

true if the matrices are equal, within the tolerance limit, or false otherwise.

▲ See Also

Reference

CSMat Class Equals Overload Microsoft.Research.Liquid Namespace

CSMatFilled Method

Return indicies of filled entries in the sparse matrix (may include zeros)

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Return Value Type: IEnumerableTupleInt32, Int32 Sequence of x,y tuples to access matrix with

⊿ See Also

Reference CSMat Class Microsoft.Research.Liquid Namespace

CSMatKron Method

Overload List

	Name	Description
= 0	Kron(Int32)	Computes the Kronecker Product of this matrix with an identity matrix. The identity matrix is on the right-hand side of the product.
.≓ Q	Kron(CSMat)	Computes the Kronecker Product of this matrix with another matrix.
Тор		

⊿ See Also

Reference CSMat Class Microsoft.Research.Liquid Namespace

CSMatKron Method (Int32)

Computes the Kronecker Product of this matrix with an identity matrix. The identity matrix is on the right-hand side of the product.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

lenl

Type: SystemInt32 The dimension of the identity matrix (N of NxN).

Return Value Type: CSMat The result of the Kronecker product

▲ See Also

Reference CSMat Class Kron Overload Microsoft.Research.Liquid Namespace

CSMatKron Method (CSMat)

Computes the Kronecker Product of this matrix with another matrix.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

m2

Type: Microsoft.Research.LiquidCSMat The right-hand side matrix in the Kronecker product.

Return Value

Type: CSMat The result of the Kronecker product

⊿ See Also

Reference CSMat Class Kron Overload Microsoft.Research.Liquid Namespace

CSMatMul Method

Overload List

		Name	Description
	≡0	Mul(CSMat)	Computes the product of this matrix and another matrix. Highly efficient.
	≓ ©	Mul(CVec)	Computes the product of this matric and a vector.
	Тор		
4	See Also	D	
	Reference CSMat Class Microsoft.Research.Liquid Namespace		

CSMatMul Method (CSMat)

Computes the product of this matrix and another matrix. Highly efficient.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

m2

Type: Microsoft.Research.LiquidCSMat The right-hand matrix to be multiplied by this matrix.

Return Value Type: CSMat New resulting sparse matrix

J See Also

Reference CSMat Class Mul Overload Microsoft.Research.Liquid Namespace

CSMatMul Method (CVec)

Computes the product of this matric and a vector.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

V

Type: Microsoft.Research.LiquidCVec The vector to multiply.

Return Value Type: CVec The resulting vector

⊿ See Also

Reference CSMat Class Mul Overload Microsoft.Research.Liquid Namespace

CSMatRead Method

Reads a matrix from a stream. The matrix must originally have been written using the Write method.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#
static member Read :
 br : BinaryReader -> CSMat

Copy

Parameters

br

Type: System.IOBinaryReader The stream to read the data from.

Return Value Type: CSMat The read-in matrix

▲ See Also

Reference CSMat Class Microsoft.Research.Liquid Namespace

CSMatToString Method

Gets a string representation of this matrix. Note that all non-zero elements are listed, so this string may be quite long.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Copy _

abstract ToString : unit -> string
override ToString : unit -> string

Return Value Type: String The string representation

⊿ See Also

Reference CSMat Class Microsoft.Research.Liquid Namespace

CSMatUnitaryError Method

Performs a rough check to see if this matrix is actually unitary. It calculates the biggest deviation from 1 of the diagonal elements of (this adj)*this. In other words, it calculates the largest deviation of the length of a row, viewed as a complex vector, from 1. The orthogonality of different rows is not checked.

Namespace: Microsoft.Research.Liquid

Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

member UnitaryError : unit -> float

Return Value Type: Double The worst deviation from 1.

▲ See Also

Reference CSMat Class Microsoft.Research.Liquid Namespace

CSMatWrite Method

Writes this matrix to a stream. The matrix may be recreated by using the Read method.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____Copy _____COPA ____COPA ____COPA ____COPA ____COPA ____COPA ____COPA ___C

Parameters

bw

Type: System.IOBinaryWriter The stream to serialize this matrix to

⊿ See Also

Reference CSMat Class Microsoft.Research.Liquid Namespace

CSMat Operators

The CSMat type exposes the following members.

▲ Operators

	Name	Description
<u>(∕</u> =+) S	Multiply(CSMat, CSMat)	Computes the product of two matrices. Highly efficient.
(/- =+) S	Multiply(CSMat, CVec)	Computes the product of a matrix and a vector.
(<u>/-</u> =+) S	MultiplyBang	Computes the Kronecker product of two matrices

Тор

▲ See Also

Reference CSMat Class Microsoft.Research.Liquid Namespace

CSMatMultiply Operator

Overload List

	Name	Description	
⊻ <u>-</u> =+) S	Multiply(CSMat, CSMat)	Computes the product of two matrices. Highly efficient.	
⊻ <u>−</u> =+) S	Multiply(CSMat, CVec)	Computes the product of a matrix and a vector.	
Тор			
⊿ See Also			
Refere CSMat Microso	nce Class oft.Research.Liquid Nam	nespace	

CSMatMultiply Operator (CSMat, CSMat)

Computes the product of two matrices. Highly efficient.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

```
F#

static let inline (*)

m1 : CSMat *

m2 : CSMat : CSMat
```

Parameters

m1

Type: Microsoft.Research.LiquidCSMat The left-hand matrix.

m2

Type: Microsoft.Research.LiquidCSMat THe right-hand matrix.

Return Value

Type: CSMat The resulting product matrix

⊿ See Also

Reference CSMat Class Multiply Overload Microsoft.Research.Liquid Namespace

CSMatMultiply Operator (CSMat, CVec)

Computes the product of a matrix and a vector.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

```
static let inline (*)
    m : CSMat *
    v : CVec : CVec
```

Parameters

т

Type: Microsoft.Research.LiquidCSMat The matrix

V

Type: Microsoft.Research.LiquidCVec The vector

Return Value Type: CVec The resulting vector

⊿ See Also

Reference CSMat Class Multiply Overload Microsoft.Research.Liquid Namespace

CSMatMultiplyBang Operator

Computes the Kronecker product of two matrices

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

_ Copy _

F# does not support this operator.

Parameters

m1

Type: Microsoft.Research.LiquidCSMat The left matrix

m2

Type: Microsoft.Research.LiquidCSMat The right matrix

Return Value Type: CSMat The result of the Kronecker product

⊿ See Also

Reference CSMat Class Microsoft.Research.Liquid Namespace

CVec Class

A block-sparse vector of complex numbers.

▲ Inheritance Hierarchy

SystemObject Microsoft.Research.LiquidCVec

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

⊿ Syntax

F#

Copy _

```
[<SerializableAttribute>]
type CVec = class end
```

The CVec type exposes the following members.

Constructors

	Name	Description
= \$	CVec(UInt64, Boolean)	Creates a new vector filled with zeroes.
=∳	CVec(Double, FSharpOptionDouble, FSharpOptionBoolean)	Creates a new vector from initial value vectors, real and imaginary. Note that the length of the new vector must be no more than 2^20.
- =	CVec(UInt64,	Creates a new vector. By

FSharpOptionDouble,	default, the vector is all
FSharpOptionDouble,	zero. Optionally, the real
FSharpOptionBoolean)	and impaginary initial
	values may be provided;
	in this case, the vector
	must be no more than
	2 ²⁰ in length.

Тор

▲ Properties

	Name	Description
	iSafe	Gets the imaginary part of an element of this vector by index.
	Length	The length of this vector
*	rSafe	Gets the real part of an element of this vector by index.
	Safe	Gets an element of this vector by index.

Тор

▲ Methods

	Name	Description
≕ ∲	AddMCC	Calculates the norm of this vector. The norm is the square root of the sum of the complex magnitudes of the vector elements.
. ⊒ ∲	Сору	Creates an exact copy of this vector.

	Dump	Dumps this vector using the provided function.
.≓ ©	DumpMCC	Dump vector MCC with provided function
≡	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
≓ ©	GetHashCode	Serves as the default hash function. (Inherited from Object.)
≓ ©	GetType	Gets the Type of the current instance. (Inherited from Object.)
≓ \$	GlobalPhase	Estimates the global phase of a vector that represents a quantum state. The estimate is computed as the complex phase of the vector element with the greatest amplitude.
≓Q	Kron	Computes the Kronecker product of this vector and another.
÷	NonZeros	Gets non-zero indices upto a max count
. ≡ ©	Normalize	Normalizes this vector to length 1.0.
≓ ≬ S	Read	Reads a new vector from a binary file. Note that this routine is only intended to read vectors written

			with the Write method.
	≓ \$	Scale	Scale all elements of the vector (unsafe)
	=∲	ToString	Converts this vector to a string. At most the first 512 entries are displayed. (Overrides ObjectToString.)
	- E	Write	Writes this vector to a binary file.
	= Q	Zero	Zeroes this vector. Note that this also densely fills in the vector.
	Тор		
4	Operato	ors	
		Name	Description
	<u>(/-</u> =+) S	MultiplyBang	Computes the Kronecker product of two vectors.
	Тор		
4	See Als	0	
	Reference Microsoft.Research.Liquid Namespace		

CVec Constructor

Overload List

	Name	Description
≓∲	CVec(UInt64, Boolean)	Creates a new vector filled with zeroes.
≓	CVec(Double, FSharpOptionDouble, FSharpOptionBoolean)	Creates a new vector from initial value vectors, real and imaginary. Note that the length of the new vector must be no more than 2^20.
. ≓	CVec(UInt64, FSharpOptionDouble, FSharpOptionDouble, FSharpOptionBoolean)	Creates a new vector. By default, the vector is all zero. Optionally, the real and impaginary initial values may be provided; in this case, the vector must be no more than 2^20 in length.

Тор

⊿ See Also

Reference CVec Class Microsoft.Research.Liquid Namespace

CVec Constructor (UInt64, Boolean)

Creates a new vector filled with zeroes.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

_len

Type: SystemUInt64 The length of the vector

force

Type: SystemBoolean

Option to force complete (non-sparse) allocation of the vector. The default to false, which leaves the vector sparse.

⊿ See Also

Reference CVec Class CVec Overload Microsoft.Research.Liquid Namespace

CVec Constructor (Double, FSharpOptionDouble, FSharpOptionBoolean)

Creates a new vector from initial value vectors, real and imaginary. Note that the length of the new vector must be no more than 2^20.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

rs

Type: SystemDouble The vector of real initial values

is

Type: Microsoft.FSharp.CoreFSharpOptionDouble

An optional vector of imaginary initial values; default is zero *force*

Type: **Microsoft.FSharp.CoreFSharpOption**Boolean Option to force complete (non-sparse) allocation of the vector; defaults to false

⊿ See Also

Reference

CVec Class CVec Overload Microsoft.Research.Liquid Namespace

CVec Constructor (UInt64, FSharpOptionDouble, FSharpOptionDouble, FSharpOptionBoolean)

Creates a new vector. By default, the vector is all zero. Optionally, the real and impaginary initial values may be provided; in this case, the vector must be no more than 2^20 in length.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# Copy
new :
    __len : uint64 *
    rsInit : FSharpOption<float[]> *
    isInit : FSharpOption<float[]> *
    forceAlloc : FSharpOption<bool> -> CVec
Parameters
len
```

Type: SystemUInt64

The length of the vector

rsInit

Type: Microsoft.FSharp.CoreFSharpOptionDouble

Optional real parts of the initial values; if provided, there must be an entry for each element in the vector. Default is zero.

isInit

Type: Microsoft.FSharp.CoreFSharpOptionDouble

Optional imaginary of the initial values; if provided, there must be an entry for each element in the vector. Default is zero. *forceAlloc*

Type: Microsoft.FSharp.CoreFSharpOptionBoolean

Option to force complete (non-sparse) allocation of the vector; defaults to false

⊿ See Also

Reference CVec Class CVec Overload Microsoft.Research.Liquid Namespace

CVec Properties

The CVec type exposes the following members.

▲ Properties

	Name	Description
₩	iSafe	Gets the imaginary part of an element of this vector by index.
**	Length	The length of this vector
*	rSafe	Gets the real part of an element of this vector by index.
*	Safe	Gets an element of this vector by index.
Тор		
See A	Also	

Reference CVec Class Microsoft.Research.Liquid Namespace

CVeciSafe Property

Gets the imaginary part of an element of this vector by index.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

member iSafe : float with get, set

Parameters

i

Type: SystemUInt64 The index of the element to get

Return Value Type: Double The imaginary part of the Complex value of the element

⊿ See Also

Reference CVec Class Microsoft.Research.Liquid Namespace

CVecLength Property

The length of this vector

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

_ Copy _

member Length : uint64 with get

Property Value Type: UInt64

⊿ See Also

Reference CVec Class Microsoft.Research.Liquid Namespace
CVecrSafe Property

Gets the real part of an element of this vector by index.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

member rSafe : float with get, set

Parameters

i

Type: SystemUInt64 The index of the element to get

Return Value Type: Double The real part of the Complex value of the element

⊿ See Also

Reference CVec Class Microsoft.Research.Liquid Namespace

CVecSafe Property

Gets an element of this vector by index.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy _

▲ Syntax

F#

member Safe : Complex with get, set

Parameters

i

Type: SystemUInt64 The index of the element to get

Return Value

Type: Complex The Complex value of the element

⊿ See Also

Reference CVec Class Microsoft.Research.Liquid Namespace

CVec Methods

The CVec type exposes the following members.

Methods

	Name	Description
≕ \$	AddMCC	Calculates the norm of this vector. The norm is the square root of the sum of the complex magnitudes of the vector elements.
-= \$	Сору	Creates an exact copy of this vector.
- =	Dump	Dumps this vector using the provided function.
-= \$	DumpMCC	Dump vector MCC with provided function
. ≕ ∲	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
≡∳	GetHashCode	Serves as the default hash function. (Inherited from Object.)
⊴∳	GetType	Gets the Type of the current instance. (Inherited from Object.)

=♥	GlobalPhase	Estimates the global phase of a vector that represents a quantum state. The estimate is computed as the complex phase of the vector element with the greatest amplitude.
≓ ≬	Kron	Computes the Kronecker product of this vector and another.
≓≬	NonZeros	Gets non-zero indices upto a max count
≓ ≬	Normalize	Normalizes this vector to length 1.0.
= ≬ S	Read	Reads a new vector from a binary file. Note that this routine is only intended to read vectors written with the Write method.
= \$	Scale	Scale all elements of the vector (unsafe)
≓∳	ToString	Converts this vector to a string. At most the first 512 entries are displayed. (Overrides ObjectToString.)
-= Q	Write	Writes this vector to a binary file.
≓∳	Zero	Zeroes this vector. Note that this also densely fills in the vector.

⊿ See Also

Reference CVec Class Microsoft.Research.Liquid Namespace

CVecAddMCC Method

Calculates the norm of this vector. The norm is the square root of the sum of the complex magnitudes of the vector elements.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

___ Сору __

▲ Syntax

F#

member AddMCC : unit -> float

Return Value Type: Double This vector's norm

⊿ See Also

Reference CVec Class Microsoft.Research.Liquid Namespace

CVecCopy Method

Creates an exact copy of this vector.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору _

member Copy : unit -> CVec

Return Value Type: CVec The new vector

⊿ See Also

Reference CVec Class Microsoft.Research.Liquid Namespace

CVecDump Method

Dumps this vector using the provided function.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# Copy member Dump : f : FSharpOption<FSharpFunc<int, FSharpFu</pre> level : FSharpOption<int> * maxNZ : FSharpOption<int> -> unit 4

Parameters

f

Type: Microsoft.FSharp.CoreFSharpOptionFSharpFuncInt32, FSharpFuncString, Unit

The optional output function to use. The default is showLogInd. *level*

Type: Microsoft.FSharp.CoreFSharpOptionInt32

The optional indentation level. The default is 0.

maxNZ

Type: Microsoft.FSharp.CoreFSharpOptionInt32

The optional maximum number of non-zeros to dump. The default is 256

⊿ See Also

Reference CVec Class Microsoft.Research.Liquid Namespace

CVecDumpMCC Method

Dump vector MCC with provided function

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# Copy
<pre>member DumpMCC : f : FSharpOption<fsharpfunc<int, fsharpfunc<int="" fsharpfunc<int,=""> * level : FSharpOption<int> * thresh : FSharpOption<float> * both : FSharpOption<bool> * maxNZ : FSharpOption<int> -> unit</int></bool></float></int></fsharpfunc<int,></pre>

Parameters

f

Type: Microsoft.FSharp.CoreFSharpOptionFSharpFuncInt32, FSharpFuncString, Unit

The optional output function to use. The default is showLogInd. *level*

Type: Microsoft.FSharp.CoreFSharpOptionInt32

The optional indentation level. The default is 0.

thresh

Type: Microsoft.FSharp.CoreFSharpOptionDouble

Output threshold (optional=tol*1.e+2)

both

```
Type: Microsoft.FSharp.CoreFSharpOptionBoolean
```

Output complex value as well (optional=false) *maxNZ*

Type: **Microsoft.FSharp.CoreFSharpOptionInt32** Max non-zeros to dump (optional=256)

⊿ See Also

Reference

CVec Class Microsoft.Research.Liquid Namespace

CVecGlobalPhase Method

Estimates the global phase of a vector that represents a quantum state. The estimate is computed as the complex phase of the vector element with the greatest amplitude.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Return Value

Type: TupleDouble, Complex

A tuple whose first element is the estimated phase angle, in radians, and whose second element is the unit-magnitude Complex number with the opposite phase.

⊿ See Also

Reference CVec Class Microsoft.Research.Liquid Namespace

CVecKron Method

Computes the Kronecker product of this vector and another.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

v2

Type: Microsoft.Research.LiquidCVec Vector to multiply this one by

Return Value Type: CVec The result vector

⊿ See Also

Reference CVec Class Microsoft.Research.Liquid Namespace

CVecNonZeros Method

Gets non-zero indices upto a max count

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

тх

Type: SystemInt32

Max length of list to return

thresh

Type: **Microsoft.FSharp.CoreFSharpOption**Double Threshold (optional=tol*1.0e+2)

Return Value Type: FSharpListUInt64 Indicies of non-zero (tolerance defined) entries

⊿ See Also

Reference CVec Class Microsoft.Research.Liquid Namespace

CVecNormalize Method

Normalizes this vector to length 1.0.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

len

Type: **Microsoft.FSharp.CoreFSharpOption**Double The optional current length, if already calculated. The default is to invoke and use the result of AddMCC

⊿ See Also

Reference CVec Class Microsoft.Research.Liquid Namespace

CVecRead Method

Reads a new vector from a binary file. Note that this routine is only intended to read vectors written with the Write method.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#
static member Read :
 br : BinaryReader -> CVec

Copy

Parameters

br

Type: System.IOBinaryReader The BinaryReader from which the vector should be read.

Return Value Type: CVec The new vector

⊿ See Also

Reference CVec Class Microsoft.Research.Liquid Namespace CVecWrite(BinaryWriter)

CVecScale Method

Scale all elements of the vector (unsafe)

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

scale

Type: Microsoft.Research.LiquidComplex Complex scale factor

⊿ See Also

Reference CVec Class Microsoft.Research.Liquid Namespace

CVecToString Method

Converts this vector to a string. At most the first 512 entries are displayed.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

abstract ToString : unit -> string
override ToString : unit -> string

Return Value Type: String The string

⊿ See Also

Reference CVec Class Microsoft.Research.Liquid Namespace

CVecWrite Method

Writes this vector to a binary file.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

bw

Type: System.IOBinaryWriter The BinaryWriter that this vector should be written to.

⊿ See Also

Reference CVec Class Microsoft.Research.Liquid Namespace CVecRead(BinaryReader)

CVecZero Method

Zeroes this vector. Note that this also densely fills in the vector.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



⊿ See Also

Reference CVec Class Microsoft.Research.Liquid Namespace

CVec Operators

The CVec type exposes the following members.

4	▲ Operators				
		Name	Description		
	(<u>/-</u> =+) S	MultiplyBang	Computes the Kronecker product of two vectors.		
	Тор				
4	⊿ See Also				
Reference CVec Class Microsoft.Research.Liquid Namespace					

CVecMultiplyBang Operator

Computes the Kronecker product of two vectors.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy _

▲ Syntax

F#

F# does not support this operator.

Parameters

v1

Type: Microsoft.Research.LiquidCVec Left vector

v2

Type: Microsoft.Research.LiquidCVec Right vector

Return Value Type: CVec The result vector

⊿ See Also

Reference CVec Class Microsoft.Research.Liquid Namespace

Fermion Class

Hamiltonian simulation for fermionic systems.

Inheritance Hierarchy

```
SystemObject Microsoft.Research.LiquidHamiltonian 
Microsoft.Research.LiquidFermion
```

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



The Fermion type exposes the following members.

Constructors

	Name	Description
. ≓	Fermion(Double, Double, Int32, Int32, Int32, DictionaryString, String, FSharpFuncDouble, FSharpFuncFSharpListQubit, Unit)	Creates a new Fermion instance from a prebuilt Circuit.

≡\$	Fermion(Double, Double, Int32,	Creates a
	Int32, Int32, FSharpListTupleInt32,	new
	Int32, Double,	Fermion
	FSharpListTupleInt32, Int32, Int32,	instance
	Int32, Double, DictionaryString,	from orbit
	String,	overlap
	FSharpOptionFSharpListInt32)	integrals.

▲ Properties

	Name	Description
*	bits	The mumber of phase estimation bits. This is the bit precision plus two.
	Circs	The built circuits, in bit order.
*	currentCirc	The last circuit that was run (ungrown).
	decohereModel	The decoherence model for this Hamiltonian. (Inherited from Hamiltonian.)
*	eMax	The maximum energy for phase estimation.
**	eMin	The minimum energy for phase estimation.
	Energy	The result of phase estimation, interpreted as an energy. This will always be between eMin and eMax. This value is only available after Run has been

		called.
*	Ket	Gets the Ket vector associated with this Hamiltonian (Inherited from Hamiltonian.)
*	omega	The energy range, eMax - eMin, for phase estimation.
*	order	The trotter order for phase estimation.
	Phase	The result of phase estimation, as an angle between 0 and 2*pi. This value is only available after Run has been called.
*	trotterN	The trotter number for phase estimation.
	tTotal	The total evolution time for phase estimation. This is equal to 2*p1/omega.
	Ua	The gate function that implements a full Hamiltonian time step

▲ Methods

	Name	Description
≓Ŷ	Build	Builds either a grown circuit or an exponentiated unitary.

=∳	Clean	Cleans out temporary files.
=♥	Dump	Dumps out information on this simulator.
=♥	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
⊴ ©	GetHashCode	Serves as the default hash function. (Inherited from Object.)
÷≓Ŵ	GetType	Gets the Type of the current instance. (Inherited from Object.)
⊴ ≬ S	Load	Loads a Fermion test from a .dat file based on a script.
≓ ∳ S	LoadOrbs	Loads orbital information from an array of strings.
=♥	PhaseSetup	Sets up phase estimation for all runs.

=∳	Prep	Prepares an initial state from the provided spin orbital indices.
≓∳	Run(Boolean, FSharpOptionInt32)	Runs the simulation to obtain a phase estimate. A previous call to Build() or BuildSingle() is required.
	Run(DictionaryString, String, String)	Runs a Fermion test from a .dat file, based on a script A basic axecution trace is sent to both the console and the log. Detailed information is sent just sent to the log. See the Users Manual for details on the parameters.
≕§	Run(DictionaryString, String, String)	Runs a pre-loaded Fermion test based on a script. A basic axecution trace is sent to both the console and the log. Detailed information is sent

		just sent to the log. See the Users Manual for details on the parameters.
≕∳ S	Run(DictionaryString, String, FSharpFuncDouble, FSharpFuncFSharpListQubit, Unit, Ket)	Runs a Fermion test from a pre- built circuit and state vector. See the Users Manual for details on the parameters.
=	ToString	Returns a string that represents the current object. (Inherited from Object.)

⊿ Remarks

This class simulates second-quantized models of electrons with a state space consisting of spin-up/spin-down pairs, with both one-body and two-body interactions.

In particular, this includes second-quantized quantum chemistry, where the state pairs are molecular orbitals, and the individual states are spin orbitals. In this case, the interaction terms come from various orbital overlap integrals. See the Users Manual for more detailed information.

⊿ See Also

Reference Microsoft.Research.Liquid Namespace

Fermion Constructor

Overload List

	Name	Description
≓⊘	Fermion(Double, Double, Int32, Int32, Int32, DictionaryString, String, FSharpFuncDouble, FSharpFuncFSharpListQubit, Unit)	Creates a new Fermion instance from a prebuilt Circuit.
=♥	Fermion(Double, Double, Int32, Int32, Int32, FSharpListTupleInt32, Int32, Double, FSharpListTupleInt32, Int32, Int32, Int32, Double, DictionaryString, String, FSharpOptionFSharpListInt32)	Creates a new Fermion instance from orbit overlap integrals.

Тор

⊿ See Also

Reference Fermion Class Microsoft.Research.Liquid Namespace

Fermion Constructor (Double, Double, Int32, Int32, Int32, DictionaryString, String, FSharpFuncDouble, FSharpFuncFSharpListQubit, Unit)

Creates a new Fermion instance from a prebuilt Circuit.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# _____Copy
new :
    eMin : float *
    eMax : float *
    trotterN : int *
    bits : int *
    order : int *
    dic : Dictionary<string, string> *
    Ua : FSharpFunc<float, FSharpFunc<FSharpI
</pre>
```

Parameters

eMin

Type: SystemDouble The minimum energy to use for phase estimation. eMax

Type: SystemDouble

The maximum energy to use for phase estimation.

trotterN

Type: SystemInt32

The Trotter number to use

bits

Type: SystemInt32

The number of bits of phase estimation accuracy desired. *order*

Type: SystemInt32

The Trotter order, either 1 or 2 (first or second order).

dic

Type: System.Collections.GenericDictionaryString, String A dictionary of options. See the Remarks for details.

Ua

Type: Microsoft.FSharp.CoreFSharpFuncDouble, FSharpFuncFSharpListQubit, Unit

A gate, usually a wrapped Circuit, that implements the Hamiltonian. The gate's parameter is the time step to evolve by.

Remarks

The possible options to specify are: ... For more information, see the Users Manual.

⊿ See Also

Reference Fermion Class Fermion Overload Microsoft.Research.Liquid Namespace

Fermion Constructor (Double, Double, Int32, Int32, Int32, FSharpListTupleInt32, Int32, Double, FSharpListTupleInt32, Int32, Int32, Int32, Double, DictionaryString, String, FSharpOptionFSharpListInt32)

Creates a new Fermion instance from orbit overlap integrals.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# _____Copy
New :
    eMin : float *
    eMax : float *
    trotterN : int *
    bits : int *
    order : int *
    ij : FSharpList<Tuple<int, int, float>> '
    ijkl : FSharpList<Tuple<int, int, int, ir
    dic : Dictionary<string, string> *
    preps : FSharpOption<FSharpList<int>> ->
    /
```

Parameters

eMin

Type: SystemDouble

The minimum energy to use for phase estimation.

eMax

Type: SystemDouble

The maximum energy to use for phase estimation.

trotterN

Type: SystemInt32

The Trotter number to use

bits

Type: SystemInt32

The number of bits of phase estimation accuracy desired.

order

Type: SystemInt32

The Trotter order, either 1 or 2 (first or second order).

ij

Type: Microsoft.FSharp.CollectionsFSharpListTupleInt32, Int32, Double

A list of tuples for single-body orbital constants. Each tuple represents an Hpq term, and contains p, q, and the value of Hpq. Note that pp terms are included in this category.

ijkl

Type: Microsoft.FSharp.CollectionsFSharpListTupleInt32, Int32, Int32, Int32, Double

A list of tuples for two-body orbital constants. Each tuple represents an Hpqrs term, and contains p, q, r, s, and the value of Hpqrs. Note that pqqp and pqqr terms are included in this category.

dic

Type: System.Collections.GenericDictionaryString, String A dictionary of options. See the Remarks for details.

preps

Type: **Microsoft.FSharp.CoreFSharpOptionFSharpListInt32** An optional specification of a 1-based electron prep list, for diagonal fix-up. The default is no prep and no fix-up.

▲ Remarks

The possible options to specify are: ... For more information, see the Users Manual.

⊿ See Also

Reference Fermion Class Fermion Overload Microsoft.Research.Liquid Namespace
Fermion Properties

The Fermion type exposes the following members.

▲ Properties

	Name	Description
*	bits	The mumber of phase estimation bits. This is the bit precision plus two.
	Circs	The built circuits, in bit order.
	currentCirc	The last circuit that was run (ungrown).
*	decohereModel	The decoherence model for this Hamiltonian. (Inherited from Hamiltonian.)
*	eMax	The maximum energy for phase estimation.
*	eMin	The minimum energy for phase estimation.
	Energy	The result of phase estimation, interpreted as an energy. This will always be between eMin and eMax. This value is only available after Run has been called.
	Ket	Gets the Ket vector associated

		with this Hamiltonian (Inherited from Hamiltonian.)
*	omega	The energy range, eMax - eMin, for phase estimation.
	order	The trotter order for phase estimation.
	Phase	The result of phase estimation, as an angle between 0 and 2*pi. This value is only available after Run has been called.
*	trotterN	The trotter number for phase estimation.
*	tTotal	The total evolution time for phase estimation. This is equal to 2*p1/omega.
		—) , () , ()
	Ua	The gate function that implements a full Hamiltonian time step

⊿ See Also

Reference Fermion Class Microsoft.Research.Liquid Namespace

Fermionbits Property

The mumber of phase estimation bits. This is the bit precision plus two.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____ Copy ____ Member bits : int with get _____ Property Value

Type: Int32

⊿ See Also

Reference Fermion Class Microsoft.Research.Liquid Namespace

FermionCircs Property

The built circuits, in bit order.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

member Circs : FSharpList<Circuit> with get

Property Value Type: FSharpListCircuit

⊿ See Also

Reference Fermion Class Microsoft.Research.Liquid Namespace

FermioncurrentCirc Property

The last circuit that was run (ungrown).

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

member currentCirc : Circuit with get

Property Value Type: Circuit

⊿ See Also

Reference Fermion Class Microsoft.Research.Liquid Namespace

FermioneMax Property

The maximum energy for phase estimation.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

 F#
 Copy

 member eMax : float with get

 Property Value

 Type: Double

J See Also

Reference Fermion Class Microsoft.Research.Liquid Namespace

FermioneMin Property

The minimum energy for phase estimation.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Copy _

Property Value Type: Double

⊿ See Also

Reference Fermion Class Microsoft.Research.Liquid Namespace

FermionEnergy Property

The result of phase estimation, interpreted as an energy. This will always be between eMin and eMax. This value is only available after Run has been called.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

member Energy : float with get

Property Value Type: Double

⊿ See Also

Reference Fermion Class Microsoft.Research.Liquid Namespace

Fermionomega Property

The energy range, eMax - eMin, for phase estimation.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

member omega : float with get

Property Value Type: Double

⊿ See Also

Reference Fermion Class Microsoft.Research.Liquid Namespace

Fermionorder Property

The trotter order for phase estimation.

Namespace: Microsoft.Research.Liquid **Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

member order : int with get

Copy _

Property Value Type: Int32

▲ See Also

Reference **Fermion Class** Microsoft.Research.Liquid Namespace

FermionPhase Property

The result of phase estimation, as an angle between 0 and 2*pi. This value is only available after Run has been called.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

__ Copy _

▲ Syntax

F#

member Phase : float with get

Property Value Type: Double

⊿ See Also

Reference Fermion Class Microsoft.Research.Liquid Namespace

FermiontrotterN Property

The trotter number for phase estimation.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____Copy ____ member trotterN : int with get, set _____ Property Value Type: Int32 See Also Reference

Fermion Class Microsoft.Research.Liquid Namespace

FermiontTotal Property

The total evolution time for phase estimation. This is equal to 2*p1/omega.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

member tTotal : float with get

Property Value Type: Double

⊿ See Also

Reference Fermion Class Microsoft.Research.Liquid Namespace

FermionUa Property

The gate function that implements a full Hamiltonian time step

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Property Value Type: FSharpFuncDouble, FSharpFuncFSharpListQubit, Unit

⊿ See Also

Reference Fermion Class Microsoft.Research.Liquid Namespace

Fermion Methods

The Fermion type exposes the following members.

▲ Methods

	Name	Description
≕	Build	Builds either a grown circuit or an exponentiated unitary.
-= \$	Clean	Cleans out temporary files.
÷	Dump	Dumps out information on this simulator.
≕	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
≞ ∲	GetHashCode	Serves as the default hash function. (Inherited from Object.)
=	GetType	Gets the Type of

		the current instance. (Inherited from Object.)
≓∳ S	Load	Loads a Fermion test from a .dat file based on a script.
≓≬ S	LoadOrbs	Loads orbital information from an array of strings.
≓∳	PhaseSetup	Sets up phase estimation for all runs.
= Q	Prep	Prepares an initial state from the provided spin orbital indices.
=♥	Run(Boolean, FSharpOptionInt32)	Runs the simulation to obtain a phase estimate. A previous call to Build() or BuildSingle() is required.
≕≬	Run(DictionaryString, String, String)	Runs a Fermion test from a .dat file, based on a script A basic axecution trace is sent to both the console and the log.

		Detailed information is sent just sent to the log. See the Users Manual for details on the parameters.
	Run(DictionaryString, String, String)	Runs a pre-loaded Fermion test based on a script. A basic axecution trace is sent to both the console and the log. Detailed information is sent just sent to the log. See the Users Manual for details on the parameters.
: ∳ S	Run(DictionaryString, String, FSharpFuncDouble, FSharpFuncFSharpListQubit, Unit, Ket)	Runs a Fermion test from a pre- built circuit and state vector. See the Users Manual for details on the parameters.
≓ \$	ToString	Returns a string that represents the current object. (Inherited from Object.)

Тор

⊿ See Also

Reference

Fermion Class Microsoft.Research.Liquid Namespace

FermionBuild Method

Builds either a grown circuit or an exponentiated unitary.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

gp

Type: Microsoft.Research.LiquidGrowPars

Grow parameters for Circuit.GrowGates. If the Single flag is set to true, then a single exponentiated unitary is built.

⊿ See Also

Reference Fermion Class Microsoft.Research.Liquid Namespace

FermionClean Method

Cleans out temporary files.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# Copy

member Clean : unit -> unit

⊿ See Also

Reference Fermion Class Microsoft.Research.Liquid Namespace

FermionDump Method

Dumps out information on this simulator.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

f

Type: Microsoft.FSharp.CoreFSharpOptionFSharpFuncInt32, FSharpFuncString, Unit

The optional output function to use. The default is showLogInd. *level*

Type: **Microsoft.FSharp.CoreFSharpOptionInt32** The optional indentation level. The default is 0.

⊿ See Also

Reference

Fermion Class Microsoft.Research.Liquid Namespace

FermionLoad Method

Loads a Fermion test from a .dat file based on a script.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

```
static member Load :
    dic : Dictionary<string, string> *
    dataFile : string -> string
```

Copy

Parameters

dic

Type: System.Collections.GenericDictionaryString, String A dictionary of options. See the Remarks for details.

dataFile

Type: SystemString

A relative or full path to the .dat file to load.

Return Value

Type: String

A single, multi-line string containing orbital information for the test number specified in the option dictionary. This string is suitable for passing (as the only element of an array) to LoadOrbs.

Remarks

The possible options to specify in the dictionary are: ... For more information, see the Users Manual.

⊿ See Also

Reference

Fermion Class Microsoft.Research.Liquid Namespace

FermionLoadOrbs Method

Loads orbital information from an array of strings.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# _____Copy
static member LoadOrbs :
    dic : Dictionary<string, string> *
    data : string[] -> Tuple<FSharpList<Tuple
</pre>
```

Parameters

dic

Type: System.Collections.GenericDictionaryString, String A dictionary of options. See the Remarks for details.

data

Type: SystemString

An array of formatted strings containing orbital information. Each string corresponds to a single line in a .dat file. The string format is described in the Users Manual.

Return Value

Type: TupleFSharpListTupleInt32, Int32, Double,

FSharpListTupleInt32, Int32, Int32, Int32, Double, String, Double, Double

A tuple containing the single-orbital terms; the two-orbital terms; an informational string; and the nuclear energy. The orbital terms are in the proper format to pass to the Fermion constructor.

▲ Remarks

The possible options to specify in the dictionary are: ... For more information, see the Users Manual.

⊿ See Also

Reference Fermion Class Microsoft.Research.Liquid Namespace

FermionPhaseSetup Method

Sets up phase estimation for all runs.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

```
member PhaseSetup :
    bits : int *
    alterNoise : float *
    peType : string -> unit
```

Parameters

bits

Type: SystemInt32

The number of bits of phase estimation accuracy desired. *alterNoise*

Type: SystemDouble

Magnitude of random (additive) noise to apply to evolution angles. Set this to 0.0 for no noise.

реТуре

Type: SystemString

The type of phase estimation to perform. Possible values are:

- "circ": Basic PE on the full circuit
- "expon": Create matrix exponentiations
- "noise": Compute noise while running (uses a single Unitary)
- "file": Serialize matrix exponentation to disk
- "atan": Use arc tangent to go forward and do classical post processing

• "default": Use type that naturally goes with specified GrowPars

⊿ See Also

Reference Fermion Class Microsoft.Research.Liquid Namespace

FermionPrep Method

Prepares an initial state from the provided spin orbital indices.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

prep

Type: **Microsoft.FSharp.CollectionsFSharpList**Int32 The list of spin orbitals to mark as occupied.

Return Value

Type: String

The "prep state" with exactly the listed orbitals occupied.

⊿ See Also

Reference Fermion Class Microsoft.Research.Liquid Namespace

FermionRun Method

Overload List

	Name	Description
-=♥	Run(Boolean, FSharpOptionInt32)	Runs the simulation to obtain a phase estimate. A previous call to Build() or BuildSingle() is required.
: ₩ S	Run(DictionaryString, String, String)	Runs a Fermion test from a .dat file, based on a script A basic axecution trace is sent to both the console and the log. Detailed information is sent just sent to the log. See the Users Manual for details on the parameters.
= 0 S	Run(DictionaryString, String, String)	Runs a pre-loaded Fermion test based on a script. A basic axecution trace is sent to

		both the console and the log. Detailed information is sent just sent to the log. See the Users Manual for details on the parameters.
= ≬ S	Run(DictionaryString, String, FSharpFuncDouble, FSharpFuncFSharpListQubit, Unit, Ket)	Runs a Fermion test from a pre- built circuit and state vector. See the Users Manual for details on the parameters.

Тор

⊿ See Also

Reference Fermion Class Microsoft.Research.Liquid Namespace

FermionRun Method (Boolean, FSharpOptionInt32)

Runs the simulation to obtain a phase estimate. A previous call to Build() or BuildSingle() is required.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

```
member Run :
    single : bool *
    maxTries : FSharpOption<int> -> FSharpOpt
```

Copy

Parameters

single

Type: SystemBoolean

Whether or not this Hamiltonian is built into a single Unitary. *maxTries*

Type: Microsoft.FSharp.CoreFSharpOptionInt32

An optional maximum number of measurements to take. The default is 40.

Return Value

Type: FSharpOptionInt32

None if it succeeded, or Some (int bit position) if it failed.

J See Also

Reference Fermion Class Run Overload Microsoft.Research.Liquid Namespace

FermionRun Method (DictionaryString, String, String)

Runs a Fermion test from a .dat file, based on a script A basic axecution trace is sent to both the console and the log. Detailed information is sent just sent to the log. See the Users Manual for details on the parameters.

Namespace: Microsoft.Research.Liquid

Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F#
        Copy _
        static member Run :
            dic : Dictionary<string, string> *
            dataFile : string -> unit
```

Parameters

dic

Type: System.Collections.GenericDictionaryString, String A dictionary of options. See the Remarks for details.

dataFile

Type: SystemString

A relative or full path to the .dat file to load.

▲ Remarks

The possible options to specify in the dictionary are: ... For more information, see the Users Manual.

⊿ See Also

Reference

Fermion Class Run Overload Microsoft.Research.Liquid Namespace

FermionRun Method (DictionaryString, String, String)

Runs a pre-loaded Fermion test based on a script. A basic axecution trace is sent to both the console and the log. Detailed information is sent just sent to the log. See the Users Manual for details on the parameters.

Namespace: Microsoft.Research.Liquid

Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# _____Copy ____Copy _____Copy ____COPY _____COPY ____COPY ___COPY ___COPY ___COPY ____COPY ____COPY ____COPY ___COPY __COPY _COPY __COPY _COPY _COPY __COPY _COPY _COPY __COPY _COPY _COPY _COPY _COPY _COPY _COPY _COPY _COP
```

Parameters

dic

Type: System.Collections.GenericDictionaryString, String A dictionary of options. See the Remarks for details.

data

Type: SystemString

An array of formatted strings containing orbital information. Each string corresponds to a single line in a .dat file. The string format is described in the Users Manual.

▲ Remarks

The possible options to specify in the dictionary are: ... For more information, see the Users Manual.

⊿ See Also

Reference

Fermion Class Run Overload Microsoft.Research.Liquid Namespace
FermionRun Method (DictionaryString, String, FSharpFuncDouble, FSharpFuncFSharpListQubit, Unit, Ket)

Runs a Fermion test from a pre-built circuit and state vector. See the Users Manual for details on the parameters.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

dic

```
Type: System.Collections.GenericDictionaryString, String A dictionary of options. See the Remarks for details.
```

Ua

```
Type: Microsoft.FSharp.CoreFSharpFuncDouble,
FSharpFuncFSharpListQubit, Unit
A gate, usually a wrapped Circuit, that implements the
```

Hamiltonian. The gate's parameter is the time step to evolve by.

ket

Type: Microsoft.Research.LiquidKet The state vector to use as the initial state.

Return Value

Type: TupleInt32, Fermion

A tuple of the phase estimation error bit, which will be -1 if no error occured, and the Fermion instance that was run. The Phase and Energy properties of the Fermion instance may be read to get the phase estimation results.

Remarks

The possible options to specify in the dictionary are: ... For more information, see the Users Manual.

▲ See Also

Reference Fermion Class Run Overload Microsoft.Research.Liquid Namespace

Gate Class

A quantum gate.

▲ Inheritance Hierarchy

```
SystemObject Microsoft.Research.LiquidGate
```

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

⊿ Syntax

F#

```
[<SerializableAttribute>]
type Gate = class end
```

The Gate type exposes the following members.

Constructors

		Name	Description
	= Q	Gate	Creates a new gate from scratch.
	Тор		
4	Properti	es	
		Name	Description
	*	Arity	Get arity of the gate (based on Qubits or Mat/Kraus size).
	r		

	CacheDisable	Whether or not the gate cache is disabled. The cache is used if this property is false. The cache should be disabled if gates are being built in parallel.
*	Draw	The drawing instructions for this gate.
*	Help	The help string for this gate.
	Kraus	The Kraus operator matrix list for this gate (if op type is Channel). If the gate is not a channel, this will be an empty list.
	Mat	The unitary matrix for this gate. If the gate is not defined by a single unitary, this will be a 0x0 matrix.
*	Name	The name of this gate.
	Ор	This gate's operation.
	Parent	This gate's parent, if any.
	Qubits	The arity of this gate; that is, the number of qubits the gate operates on.
*	User	The user info associated with this gate, if any.
Тор		

▲ Methods

Name

Description

=\$	AddControl	Creates a new gate by adding one or more control qubits to an existing unitary gate.
	Build	Gets the definition of a gate, using the cache. If the gate is already in the cache, the cached definition is returned. Otherwise, a new gate will be created, added to the cache, and returned.
= 0 S	CacheClear	Clears out the gate cache.
= \$	CacheStats	Gets gate cache statistics.
≓ \$	Dump	Dumps the full gate information to the consol and/or log.
=\$	Equals	Determines whether the specified object is equal to the

		current object. (Inherited from Object.)
≞	GetHashCode	Serves as the default hash function. (Inherited from Object.)
≓	GetType	Gets the Type of the current instance. (Inherited from Object.)
≡ ©	NewMat	Makes a new gate based on this gate but with new unitary matrix.
= ♥	OptimizeKraus	Optimze a Channel gate with Kraus operators (checks correctness and orders by magnitude)
= ♥	Run	Run this gate. The details depend on the current value of Ket.Mode : • RunMode : Apply this

		gate operation to the supplied qubits. • GateMode: Stores this gate definition into the current Ket. This is for internal use. • CircMode: Compiles this gate into a Circuit. This is for internal use.
⊴∳ S	ShowMem(Int32, FSharpFuncUnit, String, FSharpOptionFSharpListCircuit, FSharpOptionBoolean, FSharpOptionBoolean)	Outputs memory and garbage collection statistics to the log and console.
⊴∳ S	ShowMem(Int32, String, FSharpOptionFSharpListCircuit, FSharpOptionBoolean, FSharpOptionBoolean)	Outputs memory and garbage collection statistics to the log and console.

=0

gate helj type (Ov Obj	e's name, p string, and e. /errides jectToString.)
Тор	
₄ See Also	

Reference Microsoft.Research.Liquid Namespace

Gate Constructor

Creates a new gate from scratch.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#	Сору
new :	
	<i>Name</i> : FSharpOption <string> *</string>
	<i>Qubits</i> : FSharpOption <int> *</int>
	Mat : FSharpOption <csmat> *</csmat>
	<i>Draw</i> : FSharpOption <string> *</string>
	<i>Help</i> : FSharpOption <string> *</string>
	<i>Op</i> : FSharpOption <gateop> *</gateop>
	<pre>Parent : FSharpOption<fsharpoption<gate>></fsharpoption<gate></pre>
	<i>User</i> : FSharpOption <fsharpoption<object>></fsharpoption<object>
	<pre>Kraus : FSharpOption<fsharplist<krausop>></fsharplist<krausop></pre>

Parameters

Name

Type: Microsoft.FSharp.CoreFSharpOptionString

The optional name of the gate we're creating. The default is "". *Qubits*

Type: Microsoft.FSharp.CoreFSharpOptionInt32

The optional arity of gate; that is, the number of qubits this gate operates on. If a matrix for the gate is provided using the Mat parameter, then the arity is deduced from the dimensions of the matrix. The default value is 0.

Mat

Type: Microsoft.FSharp.CoreFSharpOptionCSMat

An optional sparse unitary matrix that implements the gate. This parameter is only used for gates that implement a unitary operation. The default is no unitary.

Draw

Type: Microsoft.FSharp.CoreFSharpOptionString

An optional string to use to render the gate. This should be a Q-Circuit drawing string, if provided. The default is "", which means that the gate doesn't show up when rendered.

Help

Type: Microsoft.FSharp.CoreFSharpOptionString

An optional help string for the gate. The default is no help string, "".

Ор

Type: Microsoft.FSharp.CoreFSharpOptionGateOp

The optional gate operation to perform. The default is Normal, which is a unitary gate defined by a matrix.

Parent

Type: **Microsoft.FSharp.CoreFSharpOptionFSharpOption**Gate The optional parent gate, if required by the gate operation type. The default is None.

User

Type: Microsoft.FSharp.CoreFSharpOptionFSharpOptionObje An optional user-defined field.

Kraus

Type: Microsoft.FSharp.CoreFSharpOptionFSharpListKrausOf An optional list of Kraus tags and matrices for Channels.

▲ See Also

Reference Gate Class Microsoft.Research.Liquid Namespace

Gate Properties

The Gate type exposes the following members.

▲ Properties

	Name	Description
**	Arity	Get arity of the gate (based on Qubits or Mat/Kraus size).
🖹 S	CacheDisable	Whether or not the gate cache is disabled. The cache is used if this property is false. The cache should be disabled if gates are being built in parallel.
*	Draw	The drawing instructions for this gate.
	Help	The help string for this gate.
**	Kraus	The Kraus operator matrix list for this gate (if op type is Channel). If the gate is not a channel, this will be an empty list.
**	Mat	The unitary matrix for this gate. If the gate is not defined by a single unitary, this will be a 0x0 matrix.
*	Name	The name of this gate.
*	Ор	This gate's operation.

	Parent	This gate's parent, if any.
*	Qubits	The arity of this gate; that is, the number of qubits the gate operates on.
	User	The user info associated with this gate, if any.
Тор		
See /	Also	
Poforo	nco	

Gate Class Microsoft.Research.Liquid Namespace

GateArity Property

Get arity of the gate (based on Qubits or Mat/Kraus size).

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____ Copy _ member Arity : int with get Property Value Type: Int32

⊿ See Also

Reference Gate Class Microsoft.Research.Liquid Namespace

GateCacheDisable Property

Whether or not the gate cache is disabled. The cache is used if this property is false. The cache should be disabled if gates are being built in parallel.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____Copy ______Copy _____Copy ____COPY _____COPY ____COPY ___COPY ___COPY ___COPY ____COPY ____COPY ____COPY ___COPY ___COPY ___COPY ___COPY ___COPY __COPY __COPY ___COPY ___COPY __COPY __COPY __COPY __COPY __COPY __COPY __COPY __COPY _COPY __COPY __COPY _COPY __COPY __COPY _COPY __COPY __COPY _COPY _COPY

Property Value Type: Boolean

⊿ See Also

Reference Gate Class Microsoft.Research.Liquid Namespace

GateDraw Property

The drawing instructions for this gate.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

. .

Copy _

member Draw : string with get

Property Value Type: String

⊿ See Also

Reference Gate Class Microsoft.Research.Liquid Namespace

GateHelp Property

The help string for this gate.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

member Help : string with get

Property Value Type: String

⊿ See Also

Reference Gate Class Microsoft.Research.Liquid Namespace

GateKraus Property

The Kraus operator matrix list for this gate (if op type is Channel). If the gate is not a channel, this will be an empty list.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

member Kraus : FSharpList<KrausOp> with get

Property Value Type: FSharpListKrausOp

▲ See Also

Reference Gate Class Microsoft.Research.Liquid Namespace

GateMat Property

The unitary matrix for this gate. If the gate is not defined by a single unitary, this will be a 0x0 matrix.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

F# Copy member Mat : CSMat with get Property Value Type: CSMat See Also Reference Gate Class

Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)

⊿ Syntax

GateName Property

The name of this gate.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____ Copy _____ Member Name : string with get

Property Value Type: String

⊿ See Also

Reference Gate Class Microsoft.Research.Liquid Namespace

GateOp Property

This gate's operation.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# ______ copy __ member Op : GateOp with get Property Value Type: GateOp • See Also Reference Gate Class

Microsoft.Research.Liquid Namespace

GateParent Property

This gate's parent, if any.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

member Parent : FSharpOption<Gate> with get

Property Value Type: FSharpOptionGate

⊿ See Also

Reference Gate Class Microsoft.Research.Liquid Namespace

GateQubits Property

The arity of this gate; that is, the number of qubits the gate operates on.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

___ Copy _

▲ Syntax

F#

member Qubits : int with get

Property Value Type: Int32

⊿ See Also

Reference Gate Class Microsoft.Research.Liquid Namespace

GateUser Property

The user info associated with this gate, if any.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# Copy member User : FSharpOption<Object> with get Property Value Type: FSharpOptionObject • See Also Reference Gate Class

Microsoft.Research.Liquid Namespace

Gate Methods

The Gate type exposes the following members.

Methods

	Name	Description
≓∲	AddControl	Creates a new gate by adding one or more control qubits to an existing unitary gate.
	Build	Gets the definition of a gate, using the cache. If the gate is already in the cache, the cached definition is returned. Otherwise, a new gate will be created, added to the cache, and returned.
= ≬ S	CacheClear	Clears out the gate cache.
= ≬ S	CacheStats	Gets gate cache

		statistics.
= \$	Dump	Dumps the full gate information to the consol and/or log.
≓\$	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
∃	GetHashCode	Serves as the default hash function. (Inherited from Object.)
≓\$	GetType	Gets the Type of the current instance. (Inherited from Object.)
≓\$	NewMat	Makes a new gate based on this gate but with new unitary matrix.
⊒	OptimizeKraus	Optimze a Channel gate with Kraus operators (checks

		correctness and orders by magnitude)
	Run	Run this gate. The details depend on the current value of Ket.Mode : • RunMode : • RunMode : Apply this gate operation to the supplied qubits. • GateMode : Stores this gate definition into the current Ket. This is for internal use. • CircMode : Compiles this gate into a Circuit. This is for internal use.
= ≬ S	ShowMem(Int32, FSharpFuncUnit, String, FSharpOptionFSharpListCircuit,	Outputs memory and garbage

	FSharpOptionBoolean, FSharpOptionBoolean)	collection statistics to the log and console.
⊴∳ S	ShowMem(Int32, String, FSharpOptionFSharpListCircuit, FSharpOptionBoolean, FSharpOptionBoolean)	Outputs memory and garbage collection statistics to the log and console.
-= \$	ToString	Returns a string representation of this gate, based on the gate's name, help string, and type. (Overrides ObjectToString.)
Top See A	lso	

Reference Gate Class Microsoft.Research.Liquid Namespace

GateAddControl Method

Creates a new gate by adding one or more control qubits to an existing unitary gate.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

___ Сору

member AddControl :
 count : FSharpOption<int> *
 noCache : FSharpOption<bool> -> Gate

Parameters

count

Type: Microsoft.FSharp.CoreFSharpOptionInt32

The optional number of control qubits to add. The default is a single control.

noCache

Type: Microsoft.FSharp.CoreFSharpOptionBoolean

An option to prevent caching of the new gate. The default is false, which means that the new gate should be cached. The key for the cache is a sequence of capital Cs, one for each control qubit added, followed by an underscore, '_', and then the name of the base gate.

Return Value Type: Gate The new controlled gate



Reference Gate Class Microsoft.Research.Liquid Namespace

GateBuild Method

Gets the definition of a gate, using the cache. If the gate is already in the cache, the cached definition is returned. Otherwise, a new gate will be created, added to the cache, and returned.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# _____Copy ____Copy _____Copy ____Copy _____Copy ____COPY _____COPY ____COPY ____COPY ____COPY ____COPY ___COPY ____COPY ____COPY ____COPY ____COPY ____COPY ____COPY ___COPY ___COPY ____COPY ____COPY ____COPY ____COPY ____COPY ____COPY ___COPY __COPY __COPY __COPY __COPY __COPY __COPY __COPY __COPY _COPY _COPY _COPY _COPY __COPY _COPY _COPY _COP
```

Parameters

key

Type: SystemString

The uniqur key to use to identify this gate in the cache. This may be more than just the gate name; for instance, for a rotation gate, the cache key must include the rotation angle as well as just "Rz". If the key is null or "", the cache will be bypassed.

gen

Type: Microsoft.FSharp.CoreFSharpFuncUnit, Gate

A function that may be used to create a definition for the gate if it is not already in the cache.

Return Value

Type: Gate

The gate associated with the provided key

⊿ See Also

Reference Gate Class Microsoft.Research.Liquid Namespace

GateCacheClear Method

Clears out the gate cache.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



GateCacheStats Method

Gets gate cache statistics.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Return Value Type: TupleInt32, Int32 A tuple of the cache hit count and cache miss count.

⊿ See Also

Reference Gate Class Microsoft.Research.Liquid Namespace

GateDump Method

Dumps the full gate information to the consol and/or log.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

f

Type: Microsoft.FSharp.CoreFSharpOptionFSharpFuncInt32, FSharpFuncString, Unit

The optional output function to use. The default is showLogInd. *level*

Type: **Microsoft.FSharp.CoreFSharpOptionInt32** The optional indentation level. The default is 0.

⊿ See Also

Reference

Gate Class Microsoft.Research.Liquid Namespace

GateNewMat Method

Makes a new gate based on this gate but with new unitary matrix.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

mat

Type: Microsoft.Research.LiquidCSMat The unitary matrix for the new gate.

Return Value Type: Gate The new gate

⊿ See Also

Reference Gate Class Microsoft.Research.Liquid Namespace

GateOptimizeKraus Method

Optimze a Channel gate with Kraus operators (checks correctness and orders by magnitude)

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

g2

```
Type: Microsoft.FSharp.CoreFSharpOptionGate
Gate to append to make more complex channels
(optional=None)
```

name

Type: Microsoft.FSharp.CoreFSharpOptionString

New name for the gate (optional=K####)

idMult

Type: Microsoft.FSharp.CoreFSharpOptionInt32

If appending, how much to multiply the parent gate IDs by (optional=count of appended)

povm

Type: Microsoft.FSharp.CoreFSharpOptionString

Tag to use as POVM symbol for the new gate
(optional="POVM")

Return Value Type: FSharpFuncFSharpListQubit, Unit New Gate Function

⊿ See Also

Reference Gate Class Microsoft.Research.Liquid Namespace

GateRun Method

Run this gate. The details depend on the current value of Ket.Mode:

- **RunMode**: Apply this gate operation to the supplied qubits.
- **GateMode**: Stores this gate definition into the current Ket. This is for internal use.
- **CircMode**: Compiles this gate into a Circuit. This is for internal use.

Namespace: Microsoft.Research.Liquid

Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

qs

Type: Microsoft.FSharp.CollectionsFSharpListQubit

The Qubits for this gate to operate on. They are also used to identify the current Ket.

⊿ See Also

Reference

Gate Class Microsoft.Research.Liquid Namespace

GateShowMem Method

Overload List

	Name	Description
≓∲ S	ShowMem(Int32, FSharpFuncUnit, String, FSharpOptionFSharpListCircuit, FSharpOptionBoolean, FSharpOptionBoolean)	Outputs memory and garbage collection statistics to the log and console.
= ≬ S	ShowMem(Int32, String,	Outputs
	FSharpOptionFSharpListCircuit, FSharpOptionBoolean, FSharpOptionBoolean)	memory and garbage collection statistics to the log and console.

⊿ See Also

Reference Gate Class Microsoft.Research.Liquid Namespace

GateShowMem Method (Int32, FSharpFuncUnit, String, FSharpOptionFSharpListCircuit, FSharpOptionBoolean, FSharpOptionBoolean)

Outputs memory and garbage collection statistics to the log and console.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

seconds

Type: SystemInt32

A minimum interval between reports. If a report was displayed within more recently, then no new report is displayed.

f

Type: Microsoft.FSharp.CoreFSharpFuncUnit, String

A function that returns a string to output as a prefix to the report.

CS

Type: **Microsoft.FSharp.CoreFSharpOptionFSharpListCircuit** An optional list of Circuits to analyze. The count of Circuits in the list is reported. If the "deep" parameter is true, then the total count of gates in these Circuits is also reported. The default is an empty list, which displays counts of 0.

deep

Type: Microsoft.FSharp.CoreFSharpOptionBoolean

An option indicating whether or not to count the gates in the list of Circuits. The default is false, which indicates that the gates should not be counted and will be reported as 0.

collect

Type: **Microsoft.FSharp.CoreFSharpOption**Boolean An option indicating whether to force a garbage collection. The default is false, to not force a collection.

⊿ See Also

Reference

Gate Class ShowMem Overload Microsoft.Research.Liquid Namespace

GateShowMem Method (Int32, String,

FSharpOptionFSharpListCircuit, FSharpOptionBoolean, FSharpOptionBoolean)

Outputs memory and garbage collection statistics to the log and console.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

seconds

Type: SystemInt32

A minimum interval between reports. If a report was displayed within more recently, then no new report is displayed.

str

Type: SystemString

A string to output as a prefix to the report.

CS

Type: **Microsoft.FSharp.CoreFSharpOptionFSharpListCircuit** An optional list of Circuits to analyze. The count of Circuits in the list is reported. If the "deep" parameter is true, then the total count of gates in these Circuits is also reported. The default is an empty list, which displays counts of 0.

deep

Type: Microsoft.FSharp.CoreFSharpOptionBoolean

An option indicating whether or not to count the gates in the list of Circuits. The default is false, which indicates that the gates should not be counted and will be reported as 0.

collect

Type: **Microsoft.FSharp.CoreFSharpOption**Boolean An option indicating whether to force a garbage collection. The default is false, to not force a collection.

⊿ See Also

Reference

Gate Class ShowMem Overload Microsoft.Research.Liquid Namespace

GateToString Method

Returns a string representation of this gate, based on the gate's name, help string, and type.

Copy

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

abstract ToString : unit -> string
override ToString : unit -> string

Return Value Type: String The string representation

⊿ See Also

Reference Gate Class Microsoft.Research.Liquid Namespace

GateOp Class

Gate operation type. This is used in Gate definitions.

▲ Inheritance Hierarchy

SystemObject Microsoft.Research.LiquidGateOp

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

[<SerializableAttribute>]
type GateOp = class end

The GateOp type exposes the following members.

Methods

	Name	Description
=♥	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
=♥	GetHashCode	Serves as the default hash function. (Inherited from Object.)
≡∳	GetType	Gets the Type of the current instance.

		(Inherited from Object.)
≓∲	ToString	Creates a string representation of this gate operation type. (Overrides ObjectToString.)

Тор

Remarks

The possible values for this type are:

- **Normal**: Implements a unitary operation and is defined by a matrix.
- **Measure**: Measures a single qubit or the joint parity of a groupt of qubits.
- **Channel(sym)**: Selects a Kraus operator to run (sym is a symbol to store the selected index for POVMs)
- **Reset(b)**: Reanimates a measured qubit to the Bit value b. If b is Unknown, then the qubit is reanimated to its last measured value.
- String: Puts a label in a diagram
- **Modify(n)**: Modifies a parent gate. The modified gate takes n more wires (input qubits) than the parent gate.
- **BCOp(n,f)**: Implements a classically-controlled gate. There are two cases: if n is greater than zero, then the control is logically based on the measured values of n qubits. If n is equal to zero, then the control is logically based on the result of one or more joint parity measurements. In either case, f is the function that determines whether or not to execute the parent gate.
- WrapOp(f): Wraps multiple gates into one logical gate. This is used for multi-level circuit drawing.
- WrapHam(pqrs,f): Wraps multiple gates into a single Hamiltonian term. This is used for GrowSingle.

⊿ See Also

Reference

Microsoft.Research.Liquid Namespace

GateOp Methods

The GateOp type exposes the following members.

Methods

	Name	Description
=♥	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
-= Q	GetHashCode	Serves as the default hash function. (Inherited from Object.)
=0	GetType	Gets the Type of the current instance. (Inherited from Object.)
-= Q	ToString	Creates a string representation of this gate operation type. (Overrides ObjectToString.)

Тор

⊿ See Also

Reference GateOp Class Microsoft.Research.Liquid Namespace

GateOpToString Method

Creates a string representation of this gate operation type.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору _

abstract ToString : unit -> string
override ToString : unit -> string

Return Value Type: String The string

⊿ See Also

Reference GateOp Class Microsoft.Research.Liquid Namespace

GrowPars Class

Parameters that control circuit growth. See Circuit.Grow, Circuit.GrowGates, and Circuit.GrowSingle.

▲ Inheritance Hierarchy

SystemObject Microsoft.Research.LiquidGrowPars

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

_____Сору

[<SerializableAttribute>]
type GrowPars = class end

The GrowPars type exposes the following members.

Constructors

	Name	Description
=♥	GrowPars(FSharpOptionInt32, FSharpOptionInt32, FSharpOptionBoolean)	Creates an instance with appropriate parameters for Circuit.GrowGates, which generates a denser circuit equivalent to an existing Circuit.

GrowPars(Boolean,	Creates an
FSharpOptionInt32,	instance with
FSharpOptionInt32,	appropriate
FSharpOptionInt32,	parameters for
FSharpOptionFSharpListInt32,	Circuit.GrowSingle,
FSharpOptionInt32,	which generates a
FSharpOptionBoolean,	single unitary
FSharpOptionBoolean,	operation
FSharpOptionTupleDouble,	equivalent to an
Boolean)	existing Circuit.

Тор

▲ Properties

Name	Description
AllowDense	If true, allow dense matrices to be generated by Circuit.GrowGates.
Coalesce	For Hamiltonian circuits, a tuple of a scale value and a flag specifying whether or not to keep rotation gates around small angles. The scale value is used if the flag is false. See the Users' Guide for details.
Diff	For Fermion circuits, a list of allowed differences between spin-up and spin-down counts. Effectively, this list constrains the possible values of the total net spin of valid configurations. An empty list, [], means to allow any difference.
ECnt	For Fermion circuits, the valid total electron count (total number of

		occupied states). A value of 0 means that any number of electrons is valid.
	Half	For Fermion circuits, are the qubits ordered so that the first half are the spin-up states? The alternative is for qubits to represent alternating spin- up andn spin-down states.
*	MaxWires	The maximum wire count for a single grown gate generated by Circuit.GrowGates
	OCnt	For Fermion circuits, the number of electron states. In chemical simulations, this is the spin orbital count.
	Parity	For Fermion circuits, whether to enforce conservation of angular momentum via parity.
	Redund	For Fermion circuits, whether to ignore (set to zero) portions of the resulting unitary that don't satisfy the electron count, parity check, or net spin constraints. See the Users' Guide for details.
*	Single	If true, build a single unitary; if false, grow gates into a denser circuit.
	Skip	Count of initial qubits to skip (because they are used for phase estimation) in Circuit.GrowSingle.
*	Verbose	The logging verbosity level. Possible

values are:

- **0**: No grow logging
- 1: Final circuit logging
- 2: Full detailed logging

Тор

▲ Methods

	Name	Description
=♥	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
=0	GetHashCode	Serves as the default hash function. (Inherited from Object.)
=∳	GetType	Gets the Type of the current instance. (Inherited from Object.)
=∳	ToString	Gets a string representation of this circuit grow parameter set. (Overrides ObjectToString.)
=0	VerboseSet	Creates a new GrowPars with a different verbosity setting.

Тор

⊿ See Also

Reference Microsoft.Research.Liquid Namespace

GrowPars Constructor

▲ Overload List

	Name	Description
	GrowPars(FSharpOptionInt32, FSharpOptionInt32, FSharpOptionBoolean)	Creates an instance with appropriate parameters for Circuit.GrowGates, which generates a denser circuit equivalent to an existing Circuit.
=♥	GrowPars(Boolean, FSharpOptionInt32, FSharpOptionInt32, FSharpOptionInt32, FSharpOptionFSharpListInt32, FSharpOptionInt32, FSharpOptionBoolean, FSharpOptionBoolean, FSharpOptionTupleDouble, Boolean)	Creates an instance with appropriate parameters for Circuit.GrowSingle, which generates a single unitary operation equivalent to an existing Circuit.

Тор

⊿ See Also

Reference GrowPars Class Microsoft.Research.Liquid Namespace

GrowPars Constructor (FSharpOptionInt32, FSharpOptionInt32, FSharpOptionBoolean)

Creates an instance with appropriate parameters for Circuit.GrowGates, which generates a denser circuit equivalent to an existing Circuit.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# Copy
new :
    maxWires : FSharpOption<int> *
    verbose : FSharpOption<int> *
    allowDense : FSharpOption<bool> -> GrowPa
```

Parameters

maxWires

Type: Microsoft.FSharp.CoreFSharpOptionInt32

An optional value for MaxWires. The default is 11. *verbose*

Type: Microsoft.FSharp.CoreFSharpOptionInt32

An optional value for **Verbosity**. The default is 0. *allowDense*

Type: Microsoft.FSharp.CoreFSharpOptionBoolean

An optional value for AllowDense. The default is false.

⊿ See Also

Reference GrowPars Class GrowPars Overload Microsoft.Research.Liquid Namespace

GrowPars Constructor (Boolean, FSharpOptionInt32, FSharpOptionInt32, FSharpOptionInt32, FSharpOptionFSharpListInt32, FSharpOptionInt32, FSharpOptionBoolean, FSharpOptionBoolean, FSharpOptionTupleDouble, Boolean)

Creates an instance with appropriate parameters for Circuit.GrowSingle, which generates a single unitary operation equivalent to an existing Circuit.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



```
skip : FSharpOption<int> *
diff : FSharpOption<FSharpList<int>> *
verbose : FSharpOption<int> *
parity : FSharpOption<bool> *
redund : FSharpOption<bool> *
coalesce : FSharpOption<Tuple<float, bool</pre>
```

Parameters

half

4

Type: SystemBoolean

The value for Half. This parameter is required.

eCnt

Type: Microsoft.FSharp.CoreFSharpOptionInt32

An optional value for ECnt. The default is 0.

oCnt

Type: Microsoft.FSharp.CoreFSharpOptionInt32

An optional value for OCnt. The default is 0.

skip

Type: Microsoft.FSharp.CoreFSharpOptionInt32

An optional value for Skip. The default is 0.

diff

Type: Microsoft.FSharp.CoreFSharpOptionFSharpListInt32

An optional value for ECnt. The default is an empty list, []. *verbose*

Type: Microsoft.FSharp.CoreFSharpOptionInt32

An optional value for Verbose. The default is 1.

parity

Type: Microsoft.FSharp.CoreFSharpOptionBoolean

An optional value for Parity. The default is false. *redund*

Type: Microsoft.FSharp.CoreFSharpOptionBoolean

An optional value for Redund. The default is 0.

coalesce

Type: Microsoft.FSharp.CoreFSharpOptionTupleDouble, Boolean An optional value for Coalesce. The default is (0.0, false).

⊿ See Also

Reference GrowPars Class GrowPars Overload Microsoft.Research.Liquid Namespace

GrowPars Properties

The GrowPars type exposes the following members.

▲ Properties

	Name	Description
*	AllowDense	If true, allow dense matrices to be generated by Circuit.GrowGates.
	Coalesce	For Hamiltonian circuits, a tuple of a scale value and a flag specifying whether or not to keep rotation gates around small angles. The scale value is used if the flag is false. See the Users' Guide for details.
	Diff	For Fermion circuits, a list of allowed differences between spin-up and spin-down counts. Effectively, this list constrains the possible values of the total net spin of valid configurations. An empty list, [], means to allow any difference.
	ECnt	For Fermion circuits, the valid total electron count (total number of occupied states). A value of 0 means that any number of electrons is valid.
*	Half	For Fermion circuits, are the qubits

		ordered so that the first half are the spin-up states? The alternative is for qubits to represent alternating spin- up andn spin-down states.
*	MaxWires	The maximum wire count for a single grown gate generated by Circuit.GrowGates
	OCnt	For Fermion circuits, the number of electron states. In chemical simulations, this is the spin orbital count.
	Parity	For Fermion circuits, whether to enforce conservation of angular momentum via parity.
	Redund	For Fermion circuits, whether to ignore (set to zero) portions of the resulting unitary that don't satisfy the electron count, parity check, or net spin constraints. See the Users' Guide for details.
*	Single	If true, build a single unitary; if false, grow gates into a denser circuit.
	Skip	Count of initial qubits to skip (because they are used for phase estimation) in Circuit.GrowSingle.
	Verbose	 The logging verbosity level. Possible values are: 0: No grow logging 1: Final circuit logging 2: Full detailed logging

Тор

⊿ See Also

Reference GrowPars Class Microsoft.Research.Liquid Namespace

GrowParsAllowDense Property

If true, allow dense matrices to be generated by Circuit.GrowGates.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

member AllowDense : bool with get

Property Value Type: Boolean

⊿ See Also

Reference GrowPars Class Microsoft.Research.Liquid Namespace

GrowParsCoalesce Property

For Hamiltonian circuits, a tuple of a scale value and a flag specifying whether or not to keep rotation gates around small angles. The scale value is used if the flag is false. See the Users' Guide for details.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____Copy ____ member Coalesce : Tuple<float, bool> with get

Property Value Type: TupleDouble, Boolean

⊿ See Also

Reference GrowPars Class Microsoft.Research.Liquid Namespace

GrowParsDiff Property

For Fermion circuits, a list of allowed differences between spin-up and spin-down counts. Effectively, this list constrains the possible values of the total net spin of valid configurations. An empty list, [], means to allow any difference.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

member Diff : FSharpList<int> with get

Property Value Type: FSharpListInt32

⊿ See Also

Reference GrowPars Class Microsoft.Research.Liquid Namespace

GrowParsECnt Property

For Fermion circuits, the valid total electron count (total number of occupied states). A value of 0 means that any number of electrons is valid.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Property Value Type: Int32

⊿ See Also

Reference GrowPars Class Microsoft.Research.Liquid Namespace

GrowParsHalf Property

For Fermion circuits, are the gubits ordered so that the first half are the spin-up states? The alternative is for gubits to represent alternating spin-up andn spin-down states.

Namespace: Microsoft.Research.Liquid **Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



▲ See Also

Reference **GrowPars Class** Microsoft.Research.Liquid Namespace

GrowParsMaxWires Property

The maximum wire count for a single grown gate generated by Circuit.GrowGates

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

_ Copy _

member MaxWires : int with get

Property Value Type: Int32

⊿ See Also

Reference GrowPars Class Microsoft.Research.Liquid Namespace

GrowParsOCnt Property

For Fermion circuits, the number of electron states. In chemical simulations, this is the spin orbital count.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____ Copy ____ Member OCnt : int with get

Property Value Type: Int32

⊿ See Also

Reference GrowPars Class Microsoft.Research.Liquid Namespace
GrowParsParity Property

For Fermion circuits, whether to enforce conservation of angular momentum via parity.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

_ Copy _

▲ Syntax

F#

member Parity : bool with get

Property Value Type: Boolean

⊿ See Also

Reference GrowPars Class Microsoft.Research.Liquid Namespace

GrowParsRedund Property

For Fermion circuits, whether to ignore (set to zero) portions of the resulting unitary that don't satisfy the electron count, parity check, or net spin constraints. See the Users' Guide for details.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____ Copy member Redund : bool with get

Property Value Type: Boolean

⊿ See Also

Reference GrowPars Class Microsoft.Research.Liquid Namespace

GrowParsSingle Property

If true, build a single unitary; if false, grow gates into a denser circuit.

Copy

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

member Single : bool with get

Property Value Type: Boolean

⊿ See Also

Reference GrowPars Class Microsoft.Research.Liquid Namespace

GrowParsSkip Property

Count of initial qubits to skip (because they are used for phase estimation) in Circuit.GrowSingle.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Property Value Type: Int32

⊿ See Also

Reference GrowPars Class Microsoft.Research.Liquid Namespace

GrowParsVerbose Property

The logging verbosity level. Possible values are:

- 0: No grow logging
- 1: Final circuit logging
- 2: Full detailed logging

Namespace: Microsoft.Research.Liquid

Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Copy _

member Verbose : int with get

Property Value Type: Int32

⊿ See Also

Reference GrowPars Class Microsoft.Research.Liquid Namespace

GrowPars Methods

The GrowPars type exposes the following members.

Methods

	Name	Description
=♥	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
≕ Ø	GetHashCode	Serves as the default hash function. (Inherited from Object.)
=0	GetType	Gets the Type of the current instance. (Inherited from Object.)
=∳	ToString	Gets a string representation of this circuit grow parameter set. (Overrides ObjectToString.)
≡∳	VerboseSet	Creates a new GrowPars with a different verbosity setting.

Тор

⊿ See Also

Reference GrowPars Class Microsoft.Research.Liquid Namespace

GrowParsToString Method

Gets a string representation of this circuit grow parameter set.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

```
abstract ToString : unit -> string
override ToString : unit -> string
```

Return Value Type: String The string

⊿ See Also

Reference GrowPars Class Microsoft.Research.Liquid Namespace

GrowParsVerboseSet Method

Creates a new GrowPars with a different verbosity setting.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

vNew Type: SystemInt32

The new verbosity level

Return Value Type: GrowPars The new GrowPars instance

⊿ See Also

Reference GrowPars Class Microsoft.Research.Liquid Namespace

Hamiltonian Class

Base class for Hamiltonian dynamics simulators.

Inheritance Hierarchy

```
SystemObject Microsoft.Research.LiquidHamiltonian
Microsoft.Research.LiquidFermion
Microsoft.Research.LiquidSpin
```

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



type Hamiltonian = class end

The Hamiltonian type exposes the following members.

Constructors

		Name	Description
	÷≓Ŷ	Hamiltonian	Initializes a new instance of the Hamiltonian class
	Тор		
4	Properti	es	
		Name	Description

	decohereModel	The decoherence model for this Hamiltonian.
*	Ket	Gets the Ket vector associated with this Hamiltonian

Тор

Methods

	Name	Description
=♥	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
=♥	GetHashCode	Serves as the default hash function. (Inherited from Object.)
=∲	GetType	Gets the Type of the current instance. (Inherited from Object.)
=∳	ToString	Returns a string that represents the current object. (Inherited from Object.)

Тор

⊿ See Also

Reference Microsoft.Research.Liquid Namespace

Hamiltonian Constructor

Initializes a new instance of the Hamiltonian class

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Microsoft.Research.Liquid Namespace

Hamiltonian Properties

The Hamiltonian type exposes the following members.

▲ Properties

	Name	Description
**	decohereModel	The decoherence model for this Hamiltonian.
*	Ket	Gets the Ket vector associated with this Hamiltonian
Тор		
⊿ See A	Also	

Reference Hamiltonian Class Microsoft.Research.Liquid Namespace

HamiltoniandecohereModel Property

The decoherence model for this Hamiltonian.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# _____Copy ______
member decohereModel : FSharpList<Tuple<FSharpFur
```

Property Value

Type: FSharpListTupleFSharpFuncQubit, Unit, Double

Remarks

A decoherence model is a list of gates and a decoherence probability for each gate. Gates that do not appear have a zero decoherence probability. Note that gates appearing in this list must be fully resolved; they may have no parameters other then the list of qubits.

⊿ See Also

Reference Hamiltonian Class Microsoft.Research.Liquid Namespace

HamiltonianKet Property

Gets the Ket vector associated with this Hamiltonian

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Hamiltonian Class Microsoft.Research.Liquid Namespace

Hamiltonian Methods

The Hamiltonian type exposes the following members.

Methods

	Name	Description
=♥	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
=∳	GetHashCode	Serves as the default hash function. (Inherited from Object.)
=≬	GetType	Gets the Type of the current instance. (Inherited from Object.)
=∳	ToString	Returns a string that represents the current object. (Inherited from Object.)

Тор

⊿ See Also

Reference Hamiltonian Class Microsoft.Research.Liquid Namespace

HamiltonianGates Class

A collection of gates that are useful for Hamiltonian simulation and annealing.

▲ Inheritance Hierarchy

SystemObject Microsoft.Research.LiquidHamiltonianGates

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# _____Copy ____
[<AbstractClassAttribute>]
[<SealedAttribute>]
type HamiltonianGates = class end
```

The HamiltonianGates type exposes the following members.

Methods

	Name	Description
= ≬ S	CGtheta	Performs a controlled global phase rotation.
= ≬ S	CRx	Performs a Controlled Pauli X rotation.
⊴ ≬ S	CRy	Performs a Controlled Pauli Y rotation.

	CRz	Performs a Controlled Pauli Z rotation.
= ∲ S	CTtheta	Performs a controlled T rotation.
≓ ≬ S	Entangle	Entangles a list of qubits. This is useful for building Jordan-Wigner strings.
≓ ≬ S	Gtheta	Performs a global phase rotation. This is functionally equivalent to Rpauli (2.0*theta) I qs, but has some additional drawing options.
= ≬ S	LoadCache	Preload the cache with the ZZ, Ybasis, and YbasisAdj gates.
=ŵ S	Rpauli	Performs an arbitrary rotation based on an existing gate. The base gate may be any unitary gate with a Hermitian, idempotent matrix. Of course, all Pauli gates satisfy this criteria.
≡ŵ S	Rx	Performs a Pauli X rotation. This is functionally equivalent to Rpauli theta X qs, but has some additional drawing options.
= ≬ S	Ry	Performs a Pauli Y rotation. This is functionally equivalent to Rpauli theta Y qs, but has some additional drawing options.
= 0 S	Rz	Performs a Pauli Z rotation. This is functionally equivalent to Rpauli theta Z qs, but has some

		additional drawing options.
⊴ŵ S	Ttheta	Performs a phase gate rotation. This is functionally equivalent to Rpauli (2.0*theta) T qs, but has some additional drawing options.
⊴∳ S	UnEntangle	Unentangles a list of qubits. This is useful for building Jordan-Wigner strings.
= ≬	Ybasis	Performs a basis change from Z to Y.
≈≬ S	YbasisAdj	Performs a basis change from Y to Z. This is the adjoint of Ybasis.
≓ ≬ S	ZR	Performs a Pauli Z rotation. This is equivalent to Rpauli (2.0*theta) Z qs.
=\$ S	ZZ	Performs a ZZ gate: Pauli Zs on consectutive wires. This is used for coupling strength.
= 0 S	ZZR	Performs a Pauli ZZ rotation; that is, a simultaneous Z rotation of two qubits. This is equivalent to Rpauli (2.0*theta) ZZ qs.
Гор		
See Al	50	

Reference Microsoft.Research.Liquid Namespace

HamiltonianGates Methods

The HamiltonianGates type exposes the following members.

Methods

	Name	Description
= 0 S	CGtheta	Performs a controlled global phase rotation.
= 0 S	CRx	Performs a Controlled Pauli X rotation.
= 0 S	CRy	Performs a Controlled Pauli Y rotation.
= 0 S	CRz	Performs a Controlled Pauli Z rotation.
≓≬ S	CTtheta	Performs a controlled T rotation.
≓≬ S	Entangle	Entangles a list of qubits. This is useful for building Jordan-Wigner strings.
=\$ S	Gtheta	Performs a global phase rotation. This is functionally equivalent to Rpauli (2.0*theta) I qs, but has some additional drawing options.
= 0 S	LoadCache	Preload the cache with the ZZ, Ybasis, and YbasisAdj gates.

≓∲ S	Rpauli	Performs an arbitrary rotation based on an existing gate. The base gate may be any unitary gate with a Hermitian, idempotent matrix. Of course, all Pauli gates satisfy this criteria.
= ≬ S	Rx	Performs a Pauli X rotation. This is functionally equivalent to Rpauli theta X qs, but has some additional drawing options.
≓∳ S	Ry	Performs a Pauli Y rotation. This is functionally equivalent to Rpauli theta Y qs, but has some additional drawing options.
≓ ≬ S	Rz	Performs a Pauli Z rotation. This is functionally equivalent to Rpauli theta Z qs, but has some additional drawing options.
= ≬ S	Ttheta	Performs a phase gate rotation. This is functionally equivalent to Rpauli (2.0*theta) T qs, but has some additional drawing options.
≓Ŷ S	UnEntangle	Unentangles a list of qubits. This is useful for building Jordan-Wigner strings.
≓Ŷ S	Ybasis	Performs a basis change from Z to Y.
= ≬ S	YbasisAdj	Performs a basis change from Y to Z. This is the adjoint of Ybasis.
≓ ≬ S	ZR	Performs a Pauli Z rotation. This is

		equivalent to Rpauli (2.0*theta) Z qs.
= 0 S	ZZ	Performs a ZZ gate: Pauli Zs on consectutive wires. This is used for coupling strength.
= 0 S	ZZR	Performs a Pauli ZZ rotation; that is, a simultaneous Z rotation of two qubits. This is equivalent to Rpauli (2.0*theta) ZZ qs.

Тор

⊿ See Also

Reference HamiltonianGates Class Microsoft.Research.Liquid Namespace

HamiltonianGatesCGtheta Method

Performs a controlled global phase rotation.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

```
static member CGtheta :
    theta : float *
    factor : float *
    qs : FSharpList<Qubit> -> unit
```

Parameters

theta

Type: SystemDouble

The rotation angle. 2*pi is a full rotation.

factor

Type: SystemDouble

A multiplier, used to compute the gate label.

qs

Type: **Microsoft.FSharp.CollectionsFSharpListQubit** The first qubit in the list is the control; the second qubit is the target.

⊿ See Also

Reference HamiltonianGates Class Microsoft.Research.Liquid Namespace

HamiltonianGatesCRx Method

Performs a Controlled Pauli X rotation.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# _____Copy
static member CRx :
    theta : float *
    factor : float *
    subs : string *
    qs : FSharpList<Qubit> -> unit
```

Parameters

theta

Type: SystemDouble

The rotation angle. 4*pi is a full rotation.

factor

Type: SystemDouble

A multiplier, used to compute the gate label.

subs

Type: SystemString

A subscript to attach to the gate label. Use an empty string, "", if no subscript is desired.

qs

Type: Microsoft.FSharp.CollectionsFSharpListQubit

The first qubit in the list is the control; the second qubit is the target.

⊿ See Also

Reference

HamiltonianGates Class Microsoft.Research.Liquid Namespace

HamiltonianGatesCRy Method

Performs a Controlled Pauli Y rotation.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# _____Copy
static member CRy :
    theta : float *
    factor : float *
    subs : string *
    qs : FSharpList<Qubit> -> unit
```

Parameters

theta

Type: SystemDouble

The rotation angle. 4*pi is a full rotation.

factor

Type: SystemDouble

A multiplier, used to compute the gate label.

subs

Type: SystemString

A subscript to attach to the gate label. Use an empty string, "", if no subscript is desired.

qs

Type: Microsoft.FSharp.CollectionsFSharpListQubit

The first qubit in the list is the control; the second qubit is the target.

⊿ See Also

Reference

HamiltonianGates Class Microsoft.Research.Liquid Namespace

HamiltonianGatesCRz Method

Performs a Controlled Pauli Z rotation.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# _____Copy
static member CRz :
    theta : float *
    factor : float *
    subs : string *
    qs : FSharpList<Qubit> -> unit
```

Parameters

theta

Type: SystemDouble

The rotation angle. 4*pi is a full rotation.

factor

Type: SystemDouble

A multiplier, used to compute the gate label.

subs

Type: SystemString

A subscript to attach to the gate label. Use an empty string, "", if no subscript is desired.

qs

Type: Microsoft.FSharp.CollectionsFSharpListQubit

The first qubit in the list is the control; the second qubit is the target.

⊿ See Also

Reference

HamiltonianGates Class Microsoft.Research.Liquid Namespace

HamiltonianGatesCTtheta Method

Performs a controlled T rotation.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

```
static member CTtheta :
    theta : float *
    factor : float *
    qs : FSharpList<Qubit> -> unit
```

Parameters

theta

Type: SystemDouble

The rotation angle. 2*pi is a full rotation.

factor

Type: SystemDouble

A multiplier, used to compute the gate label.

qs

Type: **Microsoft.FSharp.CollectionsFSharpListQubit** The first qubit in the list is the control; the second qubit is the target.

⊿ See Also

Reference HamiltonianGates Class Microsoft.Research.Liquid Namespace

HamiltonianGatesEntangle Method

Entangles a list of qubits. This is useful for building Jordan-Wigner strings.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

```
static member Entangle :
    ladder : FSharpList<Tuple<int, int>> *
    qs : FSharpList<Qubit> -> unit
```

Parameters

ladder

Type: Microsoft.FSharp.CollectionsFSharpListTupleInt32, Int32

A list of pairs of qubit indices to entangle. Each entry should have a tuple of two indices that refer to qubits in the *qs* list. The entangle gate will wrap a sequence of CNOT gates, one for each tuple, each with the control being the first qubit in the tuple and the target being the second in the tuple. The CNOT gates are built in ladder list order.

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The qubits to build the ladder from.

⊿ See Also

Reference HamiltonianGates Class Microsoft.Research.Liquid Namespace
HamiltonianGatesGtheta Method

Performs a global phase rotation. This is functionally equivalent to Rpauli (2.0*theta) I qs, but has some additional drawing options.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

```
static member Gtheta :
    theta : float *
    factor : float *
    qs : FSharpList<Qubit> -> unit
```

Parameters

theta

Type: SystemDouble

The rotation angle. 2*pi is a full rotation.

factor

Type: SystemDouble

A multiplier, used to compute the gate label.

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The first qubit in the list is rotated.

⊿ See Also

Reference HamiltonianGates Class Microsoft.Research.Liquid Namespace

HamiltonianGatesLoadCache Method

Preload the cache with the ZZ, Ybasis, and YbasisAdj gates.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



⊿ See Also

Reference HamiltonianGates Class Microsoft.Research.Liquid Namespace

HamiltonianGatesRpauli Method

Performs an arbitrary rotation based on an existing gate. The base gate may be any unitary gate with a Hermitian, idempotent matrix. Of course, all Pauli gates satisfy this criteria.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

theta

Type: SystemDouble

The rotation angle. 4*pi is a full rotation.

f

Type: Microsoft.FSharp.CoreFSharpFuncFSharpListQubit, Unit

The gate to base the rotation on.

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The first qubit in the list is rotated.

⊿ See Also

Reference

HamiltonianGates Class Microsoft.Research.Liquid Namespace

HamiltonianGatesRx Method

Performs a Pauli X rotation. This is functionally equivalent to Rpauli theta X qs, but has some additional drawing options.

Copy

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

```
static member Rx :
    theta : float *
    factor : float *
    subs : string *
    qs : FSharpList<Qubit> -> unit
```

Parameters

theta

Type: SystemDouble

The rotation angle. 4*pi is a full rotation.

factor

Type: SystemDouble

A multiplier, used to compute the gate label.

subs

Type: SystemString

A subscript to attach to the gate label. Use an empty string, "", if no subscript is desired.

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The first qubit in the list is rotated.

⊿ See Also

Reference

HamiltonianGates Class Microsoft.Research.Liquid Namespace

HamiltonianGatesRy Method

Performs a Pauli Y rotation. This is functionally equivalent to Rpauli theta Y qs, but has some additional drawing options.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

```
static member Ry :
    theta : float *
    factor : float *
    subs : string *
    qs : FSharpList<Qubit> -> unit
```

Parameters

theta

Type: SystemDouble

The rotation angle. 4*pi is a full rotation.

factor

Type: SystemDouble

A multiplier, used to compute the gate label.

subs

Type: SystemString

A subscript to attach to the gate label. Use an empty string, "", if no subscript is desired.

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The first qubit in the list is rotated.

⊿ See Also

Reference

HamiltonianGates Class Microsoft.Research.Liquid Namespace

HamiltonianGatesRz Method

Performs a Pauli Z rotation. This is functionally equivalent to Rpauli theta Z qs, but has some additional drawing options.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

```
static member Rz :
    theta : float *
    factor : float *
    subs : string *
    qs : FSharpList<Qubit> -> unit
```

Parameters

theta

Type: SystemDouble

The rotation angle. 4*pi is a full rotation.

factor

Type: SystemDouble

A multiplier, used to compute the gate label.

subs

Type: SystemString

A subscript to attach to the gate label. Use an empty string, "", if no subscript is desired.

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The first qubit in the list is rotated.

⊿ See Also

Reference

HamiltonianGates Class Microsoft.Research.Liquid Namespace

HamiltonianGatesTtheta Method

Performs a phase gate rotation. This is functionally equivalent to Rpauli (2.0*theta) T qs, but has some additional drawing options.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

```
static member Ttheta :
    theta : float *
    factor : float *
    subs : string *
    qs : FSharpList<Qubit> -> unit
```

Parameters

theta

Type: SystemDouble

The rotation angle. 2*pi is a full rotation.

factor

Type: SystemDouble

A multiplier, used to compute the gate label.

subs

Type: SystemString

A subscript to attach to the gate label. Use an empty string, "", if no subscript is desired.

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The first qubit in the list is rotated.

⊿ See Also

Reference

HamiltonianGates Class Microsoft.Research.Liquid Namespace

HamiltonianGatesUnEntangle Method

Unentangles a list of qubits. This is useful for building Jordan-Wigner strings.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

```
static member UnEntangle :
    ladder : FSharpList<Tuple<int, int>> *
    qs : FSharpList<Qubit> -> unit
```

Parameters

ladder

Type: Microsoft.FSharp.CollectionsFSharpListTupleInt32, Int32

A list of pairs of qubit indices to entangle. Each entry should have a tuple of two indices that refer to qubits in the *qs* list. The entangle gate will wrap a sequence of CNOT gates, one for each tuple, each with the control being the first qubit in the tuple and the target being the second in the tuple. The CNOT gates are built in the reverse order of the ladder list.

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The qubits to build the ladder from.

⊿ See Also

Reference HamiltonianGates Class Microsoft.Research.Liquid Namespace

HamiltonianGatesYbasis Method

Performs a basis change from Z to Y.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The first qubit in the list has its basis changed.

⊿ See Also

Reference HamiltonianGates Class Microsoft.Research.Liquid Namespace

HamiltonianGatesYbasisAdj Method

Performs a basis change from Y to Z. This is the adjoint of Ybasis.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F#
static member YbasisAdj :
    qs : FSharpList<Qubit> -> unit
```

Copy

Parameters

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The first qubit in the list has its basis changed.

⊿ See Also

Reference HamiltonianGates Class Microsoft.Research.Liquid Namespace

HamiltonianGatesZR Method

Performs a Pauli Z rotation. This is equivalent to Rpauli (2.0*theta) Z qs.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

```
static member ZR :
    theta : float *
    qs : FSharpList<Qubit> -> unit
```

Parameters

theta

Type: SystemDouble

The rotation angle. 2*pi is a full rotation.

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The first qubit in the list is rotated.

⊿ See Also

Reference HamiltonianGates Class Microsoft.Research.Liquid Namespace

HamiltonianGatesZZ Method

Performs a ZZ gate: Pauli Zs on consectutive wires. This is used for coupling strength.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____Copy ____COPY _____COPY ____COPY _____COPY _____COPY _____COPY _____COPY ____COPY _____COPY ____COPY ___COPY ___COPY ___COPY ____COPY ____COPY ____COPY ___COPY __COPY ___COPY ___COPY ___COPY ___COPY __COPY __COPY __COPY __COPY __COPY __COPY __COPY _COPY _

Parameters

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The first two qubits will have Z performed on them.

⊿ See Also

Reference HamiltonianGates Class Microsoft.Research.Liquid Namespace

HamiltonianGatesZZR Method

Performs a Pauli ZZ rotation; that is, a simultaneous Z rotation of two qubits. This is equivalent to Rpauli (2.0*theta) ZZ qs.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору _

```
static member ZZR :
    theta : float *
    qs : FSharpList<Qubit> -> unit
```

Parameters

theta

Type: SystemDouble

The rotation angle. 2*pi is a full rotation.

qs

Type: Microsoft.FSharp.CollectionsFSharpListQubit The first two qubits in the list are rotated.

⊿ See Also

Reference HamiltonianGates Class Microsoft.Research.Liquid Namespace

Ket Class

Represents a state vector.

▲ Inheritance Hierarchy

```
SystemObject Microsoft.Research.LiquidKet
```

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

⊿ Syntax

F#

```
[<SerializableAttribute>]
type Ket = class end
```

The Ket type exposes the following members.

Constructors

	Name	Description
≡	Ket	Creates an empty Ket vector.
≡ ∲	Ket(FSharpOptionInt32, FSharpOptionBit)	Creates a populated Ket vector.

▲ Properties

	Count	Count of Qubits in state
	Item	Get a specific qubit in our state (by qubit ID)
**	MaxEntangled	Max entangled that we've seen (reset during Reset())
	Qubits	Get all qubits in ID order
*	RandSeed	Force the pseudo-random number gen to a known initial state
*	Rnd	Ask for a random number generator
*	Symbol	Symbol table used to store the results of joint parity measurements.
	TraceRun	Trace a circuit run (0=none 1=to log 2=to log and console, 3=Ket dumps)

Тор

▲ Methods

	Name	Description
≓∳	Add(Bit)	Adds a new qubit to the state with a classical Bit value. The new qubit is unentangled and has the provided state. Note that the new qubit is not considered measured.

. ≕ ∳	Add(CVec)	Adds a new qubit to the state with a complex state vector. The new qubit is unentangled and has the provided state.
≕	Add(Int32, Bit)	Adds multiple qubits to the state, all with the same classical Bit value. The new qubits are unentangled and have the provided state. Note that the new qubits are not considered measured.
÷	Сору	Makes a new Ket that is a deep copy of this Ket.
. ≕ ∳	Decohere	Applies a function to each Qubit in the state vector, returning the list of qubit IDs for which the function returned true. This method is usually used to apply noise to the state.
- =	Dump	Dump this state vector.
- : \$	DumpKP	Dump the portion of this state vector that includes a specific qubit and all of the qubits it is entangled with. If this Ket is a product state, then only the factor containing the specified qubit is dumped.

	Entropy	Get the entanglement entropy
≡ ©	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
≓ ©	GetHashCode	Serves as the default hash function. (Inherited from Object.)
≡©	GetType	Gets the Type of the current instance. (Inherited from Object.)
≡ ©	Join	Joins another Ket to the end of this one (added Ket is NOT usable after this)
≓∲	NormDiff	Get L2 Norm of the difference of two ket vectors
≞ ©	Prob1	Get the probability of measuring 1 for a single qubit
ΞŴ	ProbOdd	Get the probability of measuring odd parity for a set of qubits
≡©	Probs	Get the state probabilities for a list of qubits (l.e. 10 qubits)
= ()	Purity	Finds the purity of each qubit in this Ket. NOT

		OPTIMIZED.
≕∳ S	Read(BinaryReader)	Reads a new state vector from a binary stream. Note that this routine is only intended to read vectors written with the Write method.
= 0 S	Read(String)	Reads a new state vector from a file. Note that this routine is only intended to read vectors written with the Write method.
= \$	Reset	Resets this Ket to an initial state with a specified number of qubits.
≓Ŵ	Single	Treats this Ket as if all of its qubits are entangled.
=♥	ToString	Gets a string representation of this Ket. This string may be extremely long; in general, it will have 2^N lines if there are N qubits in the Ket. Generally it is better to use the Dump method instead. (Overrides ObjectToString.)
≡Ŵ	Write(BinaryWriter)	Writes this state vector to a binary stream.
≓∳	Write(String)	Writes this state vector to a file.

Тор

⊿ See Also

Reference

Microsoft.Research.Liquid Namespace

Ket Constructor

Overload List

	Name	Description	
-= Q	Ket	Creates an empty Ket vector.	
=	Ket(FSharpOptionInt32, FSharpOptionBit)	Creates a populated Ket vector.	
Тор			
⊿ See Al	See Also		
Reference Ket Class Microsoft.Research.Liquid Namespace			

Ket Constructor

Creates an empty Ket vector.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Ket Constructor (FSharpOptionInt32, FSharpOptionBit)

Creates a populated Ket vector.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

```
new :
```

```
count : FSharpOption<int> *
init : FSharpOption<Bit> -> Ket
```

Copy

Parameters

count

```
Type: Microsoft.FSharp.CoreFSharpOptionInt32
Number of qubits to allocate.
```

init

Type: Microsoft.FSharp.CoreFSharpOptionBit

An optional initial value for the qubits, either Zero or One. The default is to initialize all qubits to Zero.

⊿ See Also

Reference Ket Class Ket Overload Microsoft.Research.Liquid Namespace

Ket Properties

The Ket type exposes the following members.

▲ Properties

	Name	Description
*	Count	Count of Qubits in state
	Item	Get a specific qubit in our state (by qubit ID)
*	MaxEntangled	Max entangled that we've seen (reset during Reset())
	Qubits	Get all qubits in ID order
*	RandSeed	Force the pseudo-random number gen to a known initial state
*	Rnd	Ask for a random number generator
*	Symbol	Symbol table used to store the results of joint parity measurements.
*	TraceRun	Trace a circuit run (0=none 1=to log 2=to log and console, 3=Ket dumps)

Тор

⊿ See Also

Reference

Ket Class Microsoft.Research.Liquid Namespace

KetCount Property

Count of Qubits in state

Namespace: Microsoft.Research.Liquid **Assembly:** Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Copy member Count : int with get

Property Value Type: Int32

▲ See Also

Reference **Ket Class** Microsoft.Research.Liquid Namespace

Ketltem Property

Get a specific qubit in our state (by qubit ID)

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F# member Item : Qubit with get

Parameters

i

Type: SystemInt32

Property Value Type: Qubit

⊿ See Also

Reference Ket Class Microsoft.Research.Liquid Namespace

KetMaxEntangled Property

Max entangled that we've seen (reset during Reset())

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Copy _

member MaxEntangled : int with get

Property Value Type: Int32

⊿ See Also

Reference Ket Class Microsoft.Research.Liquid Namespace

KetQubits Property

Get all qubits in ID order

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

member Qubits : FSharpList<Qubit> with get

Property Value Type: FSharpListQubit

⊿ See Also

Reference Ket Class Microsoft.Research.Liquid Namespace
KetRandSeed Property

Force the pseudo-random number gen to a known initial state

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Microsoft.Research.Liquid Namespace

KetRnd Property

Ask for a random number generator

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Microsoft.Research.Liquid Namespace

KetSymbol Property

Symbol table used to store the results of joint parity measurements.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Microsoft.Research.Liquid Namespace

KetTraceRun Property

Trace a circuit run (0=none 1=to log 2=to log and console, 3=Ket dumps)

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

_ Сору _

member TraceRun : int with get, set

Property Value Type: Int32

⊿ See Also

Reference Ket Class Microsoft.Research.Liquid Namespace

Ket Methods

The Ket type exposes the following members.

Methods

	Name	Description
≡♥	Add(Bit)	Adds a new qubit to the state with a classical Bit value. The new qubit is unentangled and has the provided state. Note that the new qubit is not considered measured.
= \$	Add(CVec)	Adds a new qubit to the state with a complex state vector. The new qubit is unentangled and has the provided state.
≓ \$	Add(Int32, Bit)	Adds multiple qubits to the state, all with the same classical Bit value. The new qubits are unentangled and have the provided state. Note that the new qubits are not considered measured.
≓\$	Сору	Makes a new Ket that is a deep copy of this Ket.
-		

	Decohere	Applies a function to each Qubit in the state vector, returning the list of qubit IDs for which the function returned true. This method is usually used to apply noise to the state.
≡Ŵ	Dump	Dump this state vector.
≕ \$	DumpKP	Dump the portion of this state vector that includes a specific qubit and all of the qubits it is entangled with. If this Ket is a product state, then only the factor containing the specified qubit is dumped.
-= 💊	Entropy	Get the entanglement entropy
≓Ŵ	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
≓∳	GetHashCode	Serves as the default hash function. (Inherited from Object.)
≡Ŵ	GetType	Gets the Type of the current instance. (Inherited from Object.)
= Q	Join	Joins another Ket to the end of this one (added Ket is NOT usable after this)

=0	NormDiff	Get L2 Norm of the difference of two ket vectors
≓∳	Prob1	Get the probability of measuring 1 for a single qubit
≓∳	ProbOdd	Get the probability of measuring odd parity for a set of qubits
≓∳	Probs	Get the state probabilities for a list of qubits (l.e. 10 qubits)
≝∳	Purity	Finds the purity of each qubit in this Ket. NOT OPTIMIZED.
≡≬ S	Read(BinaryReader)	Reads a new state vector from a binary stream. Note that this routine is only intended to read vectors written with the Write method.
≓∳ S	Read(String)	Reads a new state vector from a file. Note that this routine is only intended to read vectors written with the Write method.
≓≬	Reset	Resets this Ket to an initial state with a specified number of qubits.
= Q	Single	Treats this Ket as if all of its

		qubits are entangled.
=♥	ToString	Gets a string representation of this Ket. This string may be extremely long; in general, it will have 2^N lines if there are N qubits in the Ket. Generally it is better to use the Dump method instead. (Overrides ObjectToString.)
≡∳	Write(BinaryWriter)	Writes this state vector to a binary stream.
≓∳	Write(String)	Writes this state vector to a file.
Тор		

⊿ See Also

Reference Ket Class Microsoft.Research.Liquid Namespace

KetAdd Method

Overload List

	Name	Description
=♥	Add(Bit)	Adds a new qubit to the state with a classical Bit value. The new qubit is unentangled and has the provided state. Note that the new qubit is not considered measured.
≡ ©	Add(CVec)	Adds a new qubit to the state with a complex state vector. The new qubit is unentangled and has the provided state.
≡ ©	Add(Int32, Bit)	Adds multiple qubits to the state, all with the same classical Bit value. The new qubits are unentangled and have the provided state. Note that the new qubits are not considered measured.

Тор

⊿ See Also

Reference Ket Class Microsoft.Research.Liquid Namespace

KetAdd Method (Bit)

Adds a new qubit to the state with a classical Bit value. The new qubit is unentangled and has the provided state. Note that the new qubit is not considered measured.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

b

Type: Microsoft.Research.LiquidBit The initial state of the added qubit.

Return Value Type: Qubit The new Qubit

⊿ See Also

Reference Ket Class Add Overload Microsoft.Research.Liquid Namespace

KetAdd Method (CVec)

Adds a new qubit to the state with a complex state vector. The new qubit is unentangled and has the provided state.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

CV

Type: Microsoft.Research.LiquidCVec The state of the added qubit.

Return Value Type: Qubit The new Qubit

⊿ See Also

Reference Ket Class Add Overload Microsoft.Research.Liquid Namespace

KetAdd Method (Int32, Bit)

Adds multiple qubits to the state, all with the same classical Bit value. The new qubits are unentangled and have the provided state. Note that the new qubits are not considered measured.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

The number of qubits to add.

b

Type: Microsoft.Research.LiquidBit The initial state of the added qubits.

Return Value Type: **FSharpListQubit** The new Qubits

▲ See Also

Reference Ket Class Add Overload Microsoft.Research.Liquid Namespace

KetCopy Method

Makes a new Ket that is a deep copy of this Ket.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

member Copy : unit -> Ket

Return Value Type: Ket The new ket vector

⊿ See Also

Reference Ket Class Microsoft.Research.Liquid Namespace

KetDecohere Method

Applies a function to each Qubit in the state vector, returning the list of qubit IDs for which the function returned true. This method is usually used to apply noise to the state.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# _____Copy ____COPY _____COPY _____COPY _____COPY _____COPY _____COPY _____COPY _____COPY _____COPY ____COPY _____COPY ____COPY ___COPY ___COPY ____COPY ____COPY ____COPY ____COPY ____COPY ___COPY ___COPY ___COPY ___COPY ___COPY ___COPY ___COPY ___CO
```

Parameters

decohere

Type: **Microsoft.FSharp.CoreFSharpFunc**Qubit, Boolean The function to apply to each qubit. This function is allowed to modify the state of the qubit.

Return Value Type: **FSharpListInt32** The list of qubits where the function returned true

⊿ See Also

Reference Ket Class Microsoft.Research.Liquid Namespace

KetDump Method

Dump this state vector.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

f

Type: Microsoft.FSharp.CoreFSharpOptionFSharpFuncInt32, FSharpFuncString, Unit

The optional output function to use. The default is showLogInd. *level*

Type: Microsoft.FSharp.CoreFSharpOptionInt32

The optional indentation level. The default is 0.

doMCC

Type: Microsoft.FSharp.CoreFSharpOptionBoolean

An option that, if true, causes probabilities (complex magnitude squared) to be output for each state entry, rather than the complex amplitude. The default is to output the complex amplitude for each state.

doSort

Type: Microsoft.FSharp.CoreFSharpOptionBoolean

An option that, if true, causes the output to be sorted in qubit ID order. The default is to sort the output.

⊿ See Also

Reference Ket Class Microsoft.Research.Liquid Namespace

KetDumpKP Method

Dump the portion of this state vector that includes a specific qubit and all of the qubits it is entangled with. If this Ket is a product state, then only the factor containing the specified qubit is dumped.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

P#
Copy
member DumpKP :
 q : Qubit *
 f : FSharpFunc<int, FSharpFunc<string, Ur
 level : int *
 doMCC : FSharpOption<bool> -> unit

Parameters

q

Type: Microsoft.Research.LiquidQubit

The Qubit that iodentifies the product state factor.

f

Type: Microsoft.FSharp.CoreFSharpFuncInt32, FSharpFuncString, Unit

The optional output function to use. The default is showLogInd. *level*

Type: SystemInt32

The optional indentation level. The default is 0.

doMCC

Type: Microsoft.FSharp.CoreFSharpOptionBoolean

An option that, if true, causes probabilities (complex magnitude

squared) to be output for each state entry, rather than the complex amplitude. The default is to output the complex amplitude for each state.

⊿ See Also

Reference Ket Class Microsoft.Research.Liquid Namespace

KetEntropy Method

Get the entanglement entropy

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# Copy
member Entropy :
    alpha : int *
    qPos : int *
    order : FSharpOption<FSharpList<int>> ->
```

Parameters

alpha

Type: SystemInt32

Denotes the generalized entropy parameter: 1 for Von Neumann or 2 for Reny

qPos

Type: SystemInt32

qubit position, 0 to nQubits (based on order)

order

Type: **Microsoft.FSharp.CoreFSharpOptionFSharpListInt32** Int list of qubit order to wind up in []=use results of previous call (optional=[0..N-1])

Return Value

Type: Double Entropy value

⊿ See Also

Reference

Ket Class Microsoft.Research.Liquid Namespace

KetJoin Method

Joins another Ket to the end of this one (added Ket is NOT usable after this)

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

kAdd

Type: Microsoft.Research.LiquidKet ket to add at the end

Return Value Type: Ket Resulting ket

⊿ See Also

Reference Ket Class Microsoft.Research.Liquid Namespace

KetNormDiff Method

Get L2 Norm of the difference of two ket vectors

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

v2

Type: Microsoft.Research.LiquidCVec Target ket vector to diff (obtained by ket.Single())

Return Value Type: Double L2 Norm of the difference

⊿ See Also

Reference Ket Class Microsoft.Research.Liquid Namespace

KetProb1 Method

Get the probability of measuring 1 for a single qubit

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

q

Type: Microsoft.Research.LiquidQubit Qubit to "fake" measure

Return Value Type: Double probability

⊿ See Also

Reference Ket Class Microsoft.Research.Liquid Namespace

KetProbOdd Method

Get the probability of measuring odd parity for a set of qubits

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F#
    Copy _
member ProbOdd :
    qs : FSharpList<Qubit> *
    basis : FSharpOption<string> -> float
```

Parameters

qs

Type: Microsoft.FSharp.CollectionsFSharpListQubit Qubits to fake measure

basis

Type: Microsoft.FSharp.CoreFSharpOptionString String of basis to use for each qubit (optional="Z")

Return Value Type: Double Probability of odd parity

⊿ See Also

Reference Ket Class Microsoft.Research.Liquid Namespace

KetProbs Method

Get the state probabilities for a list of qubits (I.e. 10 qubits)

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

qs

Type: Microsoft.FSharp.CollectionsFSharpListQubit Qubits to fake measure

Return Value Type: Double Array of state values (low bit=first qubit requested)

⊿ See Also

Reference Ket Class Microsoft.Research.Liquid Namespace

KetPurity Method

Finds the purity of each qubit in this Ket. NOT OPTIMIZED.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

_ Copy _

member Purity : unit -> float[]

Return Value Type: Double An array of purities

⊿ See Also

Reference Ket Class Microsoft.Research.Liquid Namespace

KetRead Method

Overload List

		Name	Description
	≕∳ S	Read(BinaryReader)	Reads a new state vector from a binary stream. Note that this routine is only intended to read vectors written with the Write method.
≕∳ S		Read(String)	Reads a new state vector from a file. Note that this routine is only intended to read vectors written with the Write method.
	Тор		
4	See Als	0	
	Reference Ket Class Microsoft.Res	search.Liquid Namespa	ace

KetRead Method (BinaryReader)

Reads a new state vector from a binary stream. Note that this routine is only intended to read vectors written with the Write method.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____Copy ____Copy _____Copy ____COPY _____COPY _____COPY ____COPY _____COPY ____COPY ___COPY ____COPY ____COPY ____COPY ____COPY ____COPY ____COPY ____COPY ___COPY ___COPY ___COPY ___COPY ____COPY ___COPY __COPY ___COPY ___COPY __COPY __COPY _COPY __COPY __COPY _COPY __COPY __COPY _COPY

Parameters

br

Type: System.IOBinaryReader The stream from which the vector should be read.

Return Value Type: Ket The new Ket

⊿ See Also

Reference Ket Class Read Overload Microsoft.Research.Liquid Namespace KetRead(String)

KetRead Method (String)

Reads a new state vector from a file. Note that this routine is only intended to read vectors written with the Write method.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

static member Read :
 file : string -> Ket

Parameters

file

Type: SystemString The name of the file from which the state vector should be read.

Copy

Return Value Type: Ket The new Ket

⊿ See Also

Reference Ket Class Read Overload Microsoft.Research.Liquid Namespace KetRead(BinaryReader)

KetReset Method

Resets this Ket to an initial state with a specified number of qubits.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# _____Copy
member Reset :
    want : FSharpOption<int> *
    init : FSharpOption<Bit> -> FSharpList<Qu
</pre>
```

Parameters

want

Type: Microsoft.FSharp.CoreFSharpOptionInt32

An optional new qubit count. The default is to keep the same number of qubits.

init

Type: Microsoft.FSharp.CoreFSharpOptionBit

An optional initial value for all qubits. The default is Zero.

Return Value Type: **FSharpListQubit** The new Qubits in this Ket

⊿ See Also

Reference Ket Class Microsoft.Research.Liquid Namespace

KetSingle Method

Treats this Ket as if all of its qubits are entangled.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

order

Type: **Microsoft.FSharp.CoreFSharpOptionFSharpListInt32** Optionally, a list specifying a new order for the qubits. The default is for qubits to remain in their current order.

Return Value

Type: CVec

A CVec that contains the single state vector. Note that it is unsafe to modify this vector.

⊿ See Also

Reference

Ket Class Microsoft.Research.Liquid Namespace

KetToString Method

Gets a string representation of this Ket. This string may be extremely long; in general, it will have 2^N lines if there are N qubits in the Ket. Generally it is better to use the Dump method instead.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

abstract ToString : unit -> string
override ToString : unit -> string

Return Value Type: String The string representation

⊿ See Also

Reference Ket Class Microsoft.Research.Liquid Namespace
KetWrite Method

Overload List

	Name	Description		
-= \$	Write(BinaryWriter)	Writes this state vector to a binary stream.		
-= \$	Write(String)	Writes this state vector to a file.		
Тор				
⊿ See Also				
Reference Ket Class Microsoft.Research.Liquid Namespace				

KetWrite Method (BinaryWriter)

Writes this state vector to a binary stream.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

bw

Type: System.IOBinaryWriter The stream that this Ket should be written to.

⊿ See Also

Reference Ket Class Write Overload Microsoft.Research.Liquid Namespace KetWrite(String) KetRead(BinaryReader)

KetWrite Method (String)

Writes this state vector to a file.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# _____Copy _____
member Write :
    file : string -> unit
```

Parameters

file

Type: SystemString The name of the file that this Ket should be written to.

⊿ See Also

Reference Ket Class Write Overload Microsoft.Research.Liquid Namespace Ket.Write(IO, BinaryWriter) KetRead(String)

KrausOp Class

Entries for Kraus operators in Channel Gate type

Inheritance Hierarchy

SystemObject Microsoft.Research.LiquidKrausOp

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#	Сору _			
[<sealedattribute>]</sealedattribute>				
[<serializableattribute>]</serializableattribute>				
type KrausOp =				
class				
<pre>interface IEquatable<kraus0p></kraus0p></pre>				
interface IStructuralEquatable				
end				

The KrausOp type exposes the following members.

▲ Properties

	Name	Description
*	mat	Kraus operator matrix
	tag	Name to use for POVM application (usually a user provided index)

Тор

Methods

	Name	Description
=∳	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
=♥	GetHashCode	Serves as the default hash function. (Inherited from Object.)
=♥	GetType	Gets the Type of the current instance. (Inherited from Object.)
=♥	ToString	Returns a string that represents the current object. (Inherited from Object.)

Тор

⊿ See Also

Reference Microsoft.Research.Liquid Namespace

KrausOp Properties

The KrausOp type exposes the following members.

▲ Properties

	Name	Description
*	mat	Kraus operator matrix
*	tag	Name to use for POVM application (usually a user provided index)
Тор		

⊿ See Also

Reference KrausOp Class Microsoft.Research.Liquid Namespace

KrausOpmat Property

Kraus operator matrix

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Microsoft.Research.Liquid Namespace

KrausOptag Property

Name to use for POVM application (usually a user provided index)

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



KrausOp Class Microsoft.Research.Liquid Namespace

KrausOp Methods

The KrausOp type exposes the following members.

Methods

	Name	Description
=♥	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
=0	GetHashCode	Serves as the default hash function. (Inherited from Object.)
=0	GetType	Gets the Type of the current instance. (Inherited from Object.)
=∳	ToString	Returns a string that represents the current object. (Inherited from Object.)

Тор

⊿ See Also

Reference KrausOp Class Microsoft.Research.Liquid Namespace

Noise Class

A complete noise model for a specific circuit.

▲ Inheritance Hierarchy

SystemObject Microsoft.Research.LiquidNoise

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

⊿ Syntax

F#

```
[<SerializableAttribute>]
type Noise = class end
```

The Noise type exposes the following members.

Constructors

	Name	Description
≡\$	Noise	Creates a new Noise instance.

Тор

Properties

	Name	Description
*	DampProb	The probability of amplitude- damping noise on a qubit. This allows different qubits to have

		different amplitude damping probabilities.
	DampProbs	(Set only) The amplitude damping probability for all qubits. Use DampProb to get or set the amplitude damping probaility for a single qubit.
	ECgates	The list of wrap gates that are part of error-correcting circuits, by name. A name may end with "*" to indicate a wildcard.
*	Idle	(Set only) The idle gate. By default, I (the identity) is the idle gate.
	LogGates	Whether or not to log gate execution to noise statistics during Run(). Gate execution log entires will be marked with detail="!".
	Models	The noise models from last run. This provides access to the detailed statistics.
	NoNoise	The list of noiseless gates, by name. A name may end with "*" to indicate a wildcard.
	Stats	The error statistics from the last run. Statistics are kept in reverse time order.
	TraceNoise	Whether or not to trace noise to the log as it's inserted.
*	TraceWrap	Whether or not to trace wrap gates

Тор

▲ Methods

	Name	Description
= 0 S	DefaultNoise	Creates a default noise model. The new noise model has depolarizing noise on all qubits, and all gates have unit expected duration.
= 0	Dump	Dumps noise statistics from the last run.
≓∳	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
= 0	GetHashCode	Serves as the default hash function. (Inherited from Object.)
=♥	GetType	Gets the Type of the current instance. (Inherited from Object.)
≓∳	Run	Runs a circuit with this noise model.
= 0	ToString	Returns a string that represents the current object. (Inherited from Object.)

Тор

⊿ See Also

Reference

Microsoft.Research.Liquid Namespace

Noise Constructor

Creates a new Noise instance.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

circ

Type: Microsoft.Research.LiquidCircuit The circuit to apply noise to.

ket

Type: Microsoft.Research.LiquidKet

The Ket to use for noise model.

models

Type: **Microsoft.FSharp.CollectionsFSharpList**NoiseModel A list of noise models to use, in decreasing precedence order.

▲ See Also

Reference

Noise Class Microsoft.Research.Liquid Namespace

Noise Properties

The Noise type exposes the following members.

▲ Properties

	Name	Description
	DampProb	The probability of amplitude- damping noise on a qubit. This allows different qubits to have different amplitude damping probabilities.
	DampProbs	(Set only) The amplitude damping probability for all qubits. Use DampProb to get or set the amplitude damping probaility for a single qubit.
™	ECgates	The list of wrap gates that are part of error-correcting circuits, by name. A name may end with "*" to indicate a wildcard.
*	Idle	(Set only) The idle gate. By default, I (the identity) is the idle gate.
**	LogGates	Whether or not to log gate execution to noise statistics during Run(). Gate execution log entires will be marked with detail="!".
™	Models	The noise models from last run. This provides access to the detailed

			statistics.
	*	NoNoise	The list of noiseless gates, by name. A name may end with "*" to indicate a wildcard.
	*	Stats	The error statistics from the last run. Statistics are kept in reverse time order.
	*	TraceNoise	Whether or not to trace noise to the log as it's inserted.
	*	TraceWrap	Whether or not to trace wrap gates to the log as they're executed.
	Тор		
4	See Also		
Reference Noise Class Microsoft.Research.Liquid Namespace			Namespace

NoiseDampProb Property

The probability of amplitude-damping noise on a qubit. This allows different qubits to have different amplitude damping probabilities.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy _

▲ Syntax

F#

member DampProb : float with get, set

Parameters

id

Type: SystemInt32 The ID of the qubit

Property Value Type: Double

⊿ See Also

Reference Noise Class Microsoft.Research.Liquid Namespace

NoiseDampProbs Property

(Set only) The amplitude damping probability for all qubits. Use **DampProb** to get or set the amplitude damping probaility for a single qubit.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# ______ Copy _____ Copy

Property Value Type: Double

⊿ See Also

Reference Noise Class Microsoft.Research.Liquid Namespace

NoiseECgates Property

The list of wrap gates that are part of error-correcting circuits, by name. A name may end with "*" to indicate a wildcard.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Property Value Type: FSharpListString

⊿ See Also

Reference Noise Class Microsoft.Research.Liquid Namespace

NoiseIdle Property

(Set only) The idle gate. By default, I (the identity) is the idle gate.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Reference Noise Class Microsoft.Research.Liquid Namespace

NoiseLogGates Property

Whether or not to log gate execution to noise statistics during Run(). Gate execution log entires will be marked with detail="!".

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

_ Сору _

member LogGates : bool with get, set

Property Value Type: Boolean

⊿ See Also

Reference Noise Class Microsoft.Research.Liquid Namespace

NoiseModels Property

The noise models from last run. This provides access to the detailed statistics.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору _

member Models : FSharpList<NoiseModel> with get

Property Value Type: FSharpListNoiseModel

⊿ See Also

Reference Noise Class Microsoft.Research.Liquid Namespace

NoiseNoNoise Property

The list of noiseless gates, by name. A name may end with "*" to indicate a wildcard.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Property Value Type: FSharpListString

⊿ See Also

Reference Noise Class Microsoft.Research.Liquid Namespace

NoiseStats Property

The error statistics from the last run. Statistics are kept in reverse time order.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

_ Copy _

member Stats : FSharpList<NoiseStat> with get

Property Value Type: FSharpListNoiseStat

⊿ See Also

Reference Noise Class Microsoft.Research.Liquid Namespace

NoiseTraceNoise Property

Whether or not to trace noise to the log as it's inserted.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# ______ copy __ member TraceNoise : bool with get, set Property Value Type: Boolean • See Also Reference Noise Class

Microsoft.Research.Liquid Namespace

NoiseTraceWrap Property

Whether or not to trace wrap gates to the log as they're executed.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

 F#
 Copy

 member TraceWrap : bool with get, set

 Property Value

 Type: Boolean

 J See Also

 Reference

 Noise Class

 Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)

. .

Noise Methods

The Noise type exposes the following members.

Methods

	Name	Description
≡\$ S	DefaultNoise	Creates a default noise model. The new noise model has depolarizing noise on all qubits, and all gates have unit expected duration.
= ∲	Dump	Dumps noise statistics from the last run.
≡♥	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
≡ ©	GetHashCode	Serves as the default hash function. (Inherited from Object.)
≡ ©	GetType	Gets the Type of the current instance. (Inherited from Object.)
≡ ©	Run	Runs a circuit with this noise model.
≓ ©	ToString	Returns a string that represents the current object.

Тор

⊿ See Also

Reference Noise Class Microsoft.Research.Liquid Namespace

NoiseDefaultNoise Method

Creates a default noise model. The new noise model has depolarizing noise on all qubits, and all gates have unit expected duration.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

static member DefaultNoise :
 prob : float -> NoiseModel

Parameters

prob

Type: SystemDouble The probability of depolarizing noise occuring per unit of time.

Copy

Return Value Type: NoiseModel The new noise model.

J See Also

Reference Noise Class Microsoft.Research.Liquid Namespace

NoiseDump Method

Dumps noise statistics from the last run.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

f

Type: Microsoft.FSharp.CoreFSharpOptionFSharpFuncInt32, FSharpFuncString, Unit

The optional output function to use. The default is showLogInd. *level*

Type: Microsoft.FSharp.CoreFSharpOptionInt32

The optional indentation level. The default is 0.

doStats

Type: Microsoft.FSharp.CoreFSharpOptionBoolean

An option to dump full statistics. The default is false, in which case summary statistics per gate pattern are displayed.

⊿ See Also

Reference Noise Class Microsoft.Research.Liquid Namespace

NoiseRun Method

Runs a circuit with this noise model.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

ket2

Type: Microsoft.FSharp.CoreFSharpOptionKet

An optional state vector to use to re-initialize execution. If provided, this Ket is used for the run and all times and statistics are reset. If not provided, then execution continues from the state after the last Run.

⊿ See Also

Reference Noise Class Microsoft.Research.Liquid Namespace

NoiseEvents Class

Noise statistics that are tracked for normal and error-correcting gates.

Inheritance Hierarchy

SystemObject Microsoft.Research.LiquidNoiseEvents

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

⊿ Syntax

The NoiseEvents type exposes the following members.

Properties

2

	Name	Description
*	applied	The number of times noise has been applied

	count	The number of times the model has been executed
E	events	The number of noise events

Тор

▲ Methods

	Name	Description
= \$	Accum	Increments this instance with counts from another instance.
≓ ≬ S	Default	Creates a new instance with all counts initialized to 0.
≡ ©	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
≓ ©	GetHashCode	Serves as the default hash function. (Inherited from Object.)
≡ ©	GetType	Gets the Type of the current instance. (Inherited from Object.)
÷	Reset	Resets all counters back to 0.
≞ ©	ToString	Returns a string that represents the current object. (Inherited from Object.)

Тор
⊿ See Also

Reference

Microsoft.Research.Liquid Namespace

NoiseEvents Properties

The NoiseEvents type exposes the following members.

▲ Properties

	Name	Description
	applied	The number of times noise has been applied
*	count	The number of times the model has been executed
	events	The number of noise events

Тор

⊿ See Also

Reference NoiseEvents Class Microsoft.Research.Liquid Namespace

NoiseEventsapplied Property

The number of times noise has been applied

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Copy _

member applied : int with get, set

Property Value Type: Int32

⊿ See Also

Reference NoiseEvents Class Microsoft.Research.Liquid Namespace

NoiseEventscount Property

The number of times the model has been executed

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# Copy member count : int with get, set Property Value Type: Int32

Reference NoiseEvents Class Microsoft.Research.Liquid Namespace

NoiseEventsevents Property

The number of noise events

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

 F#
 Copy

 member events : int with get, set
 Copy

 Property Value
 Type: Int32

J See Also

Reference NoiseEvents Class Microsoft.Research.Liquid Namespace

NoiseEvents Methods

The NoiseEvents type exposes the following members.

Methods

	Name	Description
≡∳	Accum	Increments this instance with counts from another instance.
= ≬ S	Default	Creates a new instance with all counts initialized to 0.
≡∳	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
≓ ∲	GetHashCode	Serves as the default hash function. (Inherited from Object.)
≓ ≬	GetType	Gets the Type of the current instance. (Inherited from Object.)
≓Q	Reset	Resets all counters back to 0.
≡∳	ToString	Returns a string that represents the current object. (Inherited from Object.)

Тор

⊿ See Also

Reference

NoiseEvents Class Microsoft.Research.Liquid Namespace

NoiseEventsAccum Method

Increments this instance with counts from another instance.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



```
y
```

Type: Microsoft.Research.LiquidNoiseEvents The instance containing counts to add to this instance's counts.

⊿ See Also

Reference NoiseEvents Class Microsoft.Research.Liquid Namespace

NoiseEventsDefault Method

Creates a new instance with all counts initialized to 0.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Copy _

static member Default : unit -> NoiseEvents

Return Value Type: NoiseEvents The new instance.

⊿ See Also

Reference NoiseEvents Class Microsoft.Research.Liquid Namespace

NoiseEventsReset Method

Resets all counters back to 0.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____Copy ____COPY _____COPY ____COPY _____COPY _____COPY _____COPY _____COPY ____COPY ___COPY ___COPY ___COPY ____COPY ____COPY ____COPY ___COPY ___COPY ___COPY ____COPY ____COPY ____COPY ___COPY __COPY ___COPY __COPY __COPY __COPY __COPY __COPY _COPY __COPY __COPY _COPY _

⊿ See Also

Reference NoiseEvents Class Microsoft.Research.Liquid Namespace

NoiseModel Class

A noise model for a particular type of gate (or set of gates).

Inheritance Hierarchy

SystemObject Microsoft.Research.LiquidNoiseModel

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

```
[<SealedAttribute>]
[<SerializableAttribute>]
type NoiseModel = class end
```

The NoiseModel type exposes the following members.

▲ Properties

	Name	Description
*	ecEvents	Accumulated noise statistics for error-correcting gates.
*	func	The noise function to execute.
	gate	The name of the gate that this noise model applies to. The name may end with an asterisk, '*', to indicate a wildcard match.

<u>-</u>

	gateEvents	Accumulated noise statistics for normal gates.
*	maxQs	The maximum number of qubits to apply noise to.
	time	The expected duration of the gate. By convention, an idle gate takes 1.0 units.

Тор

▲ Methods

	Name	Description
= \$	Default	Creates a default noise model that will apply to all gates.
=♥	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
= \$	GetHashCode	Serves as the default hash function. (Inherited from Object.)
=♥	GetType	Gets the Type of the current instance. (Inherited from Object.)
≡ ©	ToString	Returns a string that represents the current object. (Inherited from Object.)

Тор

⊿ See Also

Reference

Microsoft.Research.Liquid Namespace

NoiseModel Properties

The NoiseModel type exposes the following members.

▲ Properties

	Name	Description
	ecEvents	Accumulated noise statistics for error-correcting gates.
The second se	func	The noise function to execute.
	gate	The name of the gate that this noise model applies to. The name may end with an asterisk, '*', to indicate a wildcard match.
*	gateEvents	Accumulated noise statistics for normal gates.
*	maxQs	The maximum number of qubits to apply noise to.
	time	The expected duration of the gate. By convention, an idle gate takes 1.0 units.

Тор

▲ See Also

Reference NoiseModel Class Microsoft.Research.Liquid Namespace

NoiseModelecEvents Property

Accumulated noise statistics for error-correcting gates.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

 F#
 Copy

 member ecEvents : NoiseEvents with get

 Property Value

 Type: NoiseEvents

 J See Also

 Reference

NoiseModel Class Microsoft.Research.Liquid Namespace

NoiseModelfunc Property

The noise function to execute.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Property Value Type: FSharpFuncDouble, FSharpFuncDouble, FSharpFuncFSharpListQubit, FSharpListTupleFSharpListQubit, FSharpOptionString

⊿ See Also

Reference NoiseModel Class Microsoft.Research.Liquid Namespace

NoiseModelgate Property

The name of the gate that this noise model applies to. The name may end with an asterisk, '*', to indicate a wildcard match.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

_ Copy _

member gate : string with get

Property Value Type: String

⊿ See Also

Reference NoiseModel Class Microsoft.Research.Liquid Namespace

NoiseModelgateEvents Property

Accumulated noise statistics for normal gates.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

member gateEvents : NoiseEvents with get

Property Value Type: NoiseEvents

▲ See Also

Reference NoiseModel Class Microsoft.Research.Liquid Namespace

NoiseModelmaxQs Property

The maximum number of qubits to apply noise to.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____ Copy ____ Member maxQs : int with get _____ Property Value

Type: Int32

⊿ See Also

Reference NoiseModel Class Microsoft.Research.Liquid Namespace

NoiseModeltime Property

The expected duration of the gate. By convention, an idle gate takes 1.0 units.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

___ Copy _

▲ Syntax

F#

member time : float with get

Property Value Type: Double

▲ See Also

Reference NoiseModel Class Microsoft.Research.Liquid Namespace

NoiseModel Methods

The NoiseModel type exposes the following members.

Methods

	Name	Description
= 0 S	Default	Creates a default noise model that will apply to all gates.
=♥	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
=♥	GetHashCode	Serves as the default hash function. (Inherited from Object.)
=♥	GetType	Gets the Type of the current instance. (Inherited from Object.)
≓∳	ToString	Returns a string that represents the current object. (Inherited from Object.)

Тор

⊿ See Also

Reference NoiseModel Class Microsoft.Research.Liquid Namespace

NoiseModelDefault Method

Creates a default noise model that will apply to all gates.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

modelFunc

Type: Microsoft.FSharp.CoreFSharpFuncDouble, FSharpFuncDouble, FSharpFuncFSharpListQubit, FSharpListTupleFSharpListQubit, FSharpOptionString The noise function to apply.

Return Value

Type: NoiseModel

A noise model that will apply the given function to all gates, to a single qubit, and initialized with zero statistics.

⊿ See Also

Reference NoiseModel Class Microsoft.Research.Liquid Namespace

NoiseStat Class

Statistics tracked for each time that noise is applied.

Inheritance Hierarchy

SystemObject Microsoft.Research.LiquidNoiseStat

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

⊿ Syntax

F#

```
[<SealedAttribute>]
[<SerializableAttribute>]
type NoiseStat = class end
```

The NoiseStat type exposes the following members.

▲ Properties

	Name	Description
*	detail	User-defined noise details from the noise function output.
	dur	The duration of the noise applicationm.
*	ecGate	Whether or not thie was flagged as an error correcting gate.
*	model	The noise model that was applied.
·		

qs	The qubits that noise was applied to.
time	When noise was applied during the simulated execution.

Тор

Methods

	Name	Description
=♥	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
=0	GetHashCode	Serves as the default hash function. (Inherited from Object.)
=∲	GetType	Gets the Type of the current instance. (Inherited from Object.)
=∳	ToString	Returns a string representation of this noise statistic. (Overrides ObjectToString.)

Тор

⊿ See Also

Reference Microsoft.Research.Liquid Namespace

NoiseStat Properties

The NoiseStat type exposes the following members.

▲ Properties

	Name	Description
	detail	User-defined noise details from the noise function output.
*	dur	The duration of the noise applicationm.
*	ecGate	Whether or not thie was flagged as an error correcting gate.
*	model	The noise model that was applied.
*	qs	The qubits that noise was applied to.
*	time	When noise was applied during the simulated execution.

Тор

⊿ See Also

Reference NoiseStat Class Microsoft.Research.Liquid Namespace

NoiseStatdetail Property

User-defined noise details from the noise function output.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

member detail : string with get

Property Value Type: String

⊿ See Also

Reference NoiseStat Class Microsoft.Research.Liquid Namespace

NoiseStatdur Property

The duration of the noise applicationm.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Microsoft.Research.Liquid Namespace

NoiseStatecGate Property

Whether or not thie was flagged as an error correcting gate.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy _

▲ Syntax

F#

member ecGate : bool with get

Property Value Type: Boolean

⊿ See Also

Reference NoiseStat Class Microsoft.Research.Liquid Namespace

NoiseStatmodel Property

The noise model that was applied.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# ______ Copy __ member model : NoiseModel with get _____ Property Value Type: NoiseModel • See Also Reference

NoiseStat Class Microsoft.Research.Liquid Namespace

NoiseStatqs Property

The qubits that noise was applied to.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

 F#
 Copy

 member qs : FSharpList<Qubit> with get

 Property Value

 Type: FSharpListQubit

 J See Also

 Reference

 NoiseStat Class

Microsoft.Research.Liquid Namespace

NoiseStattime Property

When noise was applied during the simulated execution.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

 F#
 Copy

 member time : float with get

 Property Value

 Type: Double

J See Also

Reference NoiseStat Class Microsoft.Research.Liquid Namespace

NoiseStat Methods

The NoiseStat type exposes the following members.

Methods

	Name	Description
=♥	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
=∳	GetHashCode	Serves as the default hash function. (Inherited from Object.)
=0	GetType	Gets the Type of the current instance. (Inherited from Object.)
=∲	ToString	Returns a string representation of this noise statistic. (Overrides ObjectToString.)

Тор

⊿ See Also

Reference NoiseStat Class Microsoft.Research.Liquid Namespace
NoiseStatToString Method

Returns a string representation of this noise statistic.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору _

abstract ToString : unit -> string
override ToString : unit -> string

Return Value Type: String The string

⊿ See Also

Reference NoiseStat Class Microsoft.Research.Liquid Namespace

NoisyMats Class

Utility class for computing a Pauli rotation matrix. This is used to run quantum chemistry circuits with noise injected.

▲ Inheritance Hierarchy

SystemObject Microsoft.Research.LiquidNoisyMats

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____Copy [<SerializableAttribute>]

```
type NoisyMats = class end
```

The NoisyMats type exposes the following members.

Methods

	Name	Description
≡∳	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
≡ŷ S	GenRot	Generates a Pauli-based rotation matrix for an arbitrary angle.
≓ ©	GetHashCode	Serves as the default hash function.

		(Inherited from Object.)
≡\$	GetType	Gets the Type of the current instance. (Inherited from Object.)
≡∳	ToString	Returns a string that represents the current object. (Inherited from Object.)

Тор

⊿ See Also

Reference Microsoft.Research.Liquid Namespace

NoisyMats Methods

The NoisyMats type exposes the following members.

Methods

	Name	Description
=♥	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
=\$ S	GenRot	Generates a Pauli-based rotation matrix for an arbitrary angle.
=♥	GetHashCode	Serves as the default hash function. (Inherited from Object.)
=♥	GetType	Gets the Type of the current instance. (Inherited from Object.)
=♥	ToString	Returns a string that represents the current object. (Inherited from Object.)

Тор

⊿ See Also

Reference NoisyMats Class Microsoft.Research.Liquid Namespace

NoisyMatsGenRot Method

Generates a Pauli-based rotation matrix for an arbitrary angle.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____Copy ____COPY _____COPY ____COPY _____COPY ____COPY ___COPY ___COPY ___COPY ___COPY ____COPY ___COPY ___COPY ___COPY ___COPY ___COPY _

Parameters

theta

Type: SystemDouble The rotation angle.

mParent

Type: Microsoft.Research.LiquidCSMat

The base matrix for the rotation. This must be an idempotent Hermitian matrix.

Return Value

Type: CSMat A rotation matrix, exp(i*theta/2*mParent).

⊿ See Also

Reference NoisyMats Class Microsoft.Research.Liquid Namespace

Operations Class

The Operations module provides definitions of basic gates. It also includes some handy operators for manipulating qubit lists, and some operations for building gates from existing gates.

▲ Inheritance Hierarchy

SystemObject Microsoft.Research.LiquidOperations

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# _____Copy ____COPY _____COPY _____COPY ____COPY ___COPY ____COPY ____COPY ___COPY ____COPY ____COPY ____COPY ____COPY ____COPY ___C
```

The Operations type exposes the following members.

Methods

≡© ≤

	Name	Description
=\$ S	AD	Amplitude Damping Channel using two Kraus operators (always put the most probable one first)
= ∲ S	Adj	Performs the adjoint of the parent gate. This only works if the parent gate is a unitary gate.

	BC	Performs a gate under classical control. The parent gate will be executed if the control qubit has a measured value of One.
=∳ S	BCany	Performs a gate under classical control. This gate allows an arbitrary condition to be specified. The parent gate will be executed if the condition function evaluates to true.
≡ŷ S	CCgate	Performs a parent gate under two- qubit quantum control. The parent must be a unitary gate.
≡ŷ S	CCNOT	Performs a Toffoli or Controlled- Controlled-NOT gate
≓ ≬ S	Cgate	Performs a parent gate under quantum control. The parent must be a unitary gate.
≡ ∳ S	CgateNC	Performs a parent gate under quantum control. The parent must be a unitary gate. The resulting gate is not cached.
≡ŷ S	CNOT	Performs a quantum-controlled NOT gate
≡ ≬ S	CZ	Performs a quantum-controlled Pauli Z gate.
≡ŷ S	DP	Depolarizing channel using four Kraus operators (always put the most probable one first)
≈ ≬ S	Н	Performs a Hadamard gate.

≓ ≬ S	I.	Performs a Pauli I (identify) gate.
= ≬ S	JM	Performs a joint parity measurement in the given basis on a list of qubits. The result of the measurement is stored in the Ket's symbol table. The state vector is collapsed according to the result of the measurement.
≕∳ S	JMx	Performs a joint parity measurement in the X basis on a list of qubits. The result of the measurement is stored in the Ket's symbol table. The state vector is collapsed according to the result of the measurement.
≡ ŵ S	JMz	Performs a joint parity measurement in the computational basis on a list of qubits. The result of the measurement is stored in the Ket's symbol table. The state vector is collapsed according to the result of the measurement.
= ≬ S	Label	Adds a label to a circuit drawing. The label will appear above the line representing the qubit, in the center of the current column. The label will take up no logical space; it will not fill the current cell.
≓ \$ S	LabelC	Adds a label to a circuit drawing. The label be centered in and will fill the current cell.
≓Ŷ S	LabelCD	Adds a framed label to a circuit drawing. The label be centered in and will fill the current cell. The label

		will have a box drawn tightly around it.
≓Ŷ S	LabelD	Adds a label to a circuit drawing. The label will appear below the line representing the qubit, in the center of the current column. The label will take up no logical space; it will not fill the current cell.
≕∳ S	LabelL	Adds a label to a circuit drawing. The label will appear centered on the line representing the qubit, on the left side of the current column. The label will take up no logical space; it will not fill the current cell.
:∳ S	LabelR	Adds a label to a circuit drawing. The label will appear centered on the line representing the qubit, on the right side of the current column. The label will take up no logical space; it will not fill the current cell.
⊴∳ S	LabelRaw	Adds a raw label to a circuit drawing. The provided string should contain the Q-Circuit commands used to draw the label. Note that labels drawn using this gate will only appear when rendered to "qc" format.
⊴∳ S	LabelU	Adds a label to a circuit drawing. The label will appear above the line representing the qubit, in the center of the current column. The label will take up no logical space; it will not fill

		the current cell.
⊴∳ S	LoadCache	Preload the cache with basic gates: Paulis, measurement, Hadamard, phase, CNOT, Toffoli, swap, and T.
≕∳ S	Μ	Performs a measurement of a single qubit in the computational basis. The result of the measurement is stored in the measured qubit; see Qubit.Bit . The state vector is collapsed according to the result of the measurement. The measured qubit becomes "classical" and must be reset before any further quantum gates are performed on it.
= 0 S	Native	Performs any desired native operations at this point in the circuit.
⊴ ≬ S	NativeDbg	Performs any desired native operations at this point in the circuit. This version doesn't appear in a circuit drawing.
⊧∳ S	PC	Performs a gate based on a classical condition. The parent gate is executed if the test function returns true. Typically this condition is based on the results of one or more joint parity measurements.
= 0 S	R	Performs a 2pi/2^K rotation gate gate.
= 0 S	Reset	Resets a qubit to a specified initial state after it has been measured. This allows quantum gates to be

		performed against the collapsed qubit. See the M gate.
≕∳ S	Restore	Resets a qubit after it has been measured. This allows quantum gates to be performed against the collapsed qubit. The initial state of the qubit will be its last measured state. See the M gate.
≈ ≬ S	S	Performs a phase gate.
= 0 S	SWAP	Performs a swap gate, exchanging the quantum states of two qubits.
≓ ≬ S	т	Performs a pi/8 gate.
= 0 S	T_BC	Performs a transverse classically- controlled gate.
= 0 S	Transverse	Expands a parent gate to a transverse version.
=0 S	X	Performs a Pauli X gate.
= 0 S	Y	Performs a Pauli Y gate.
=0 S	Z	Performs a Pauli Z gate.

Тор

▲ Operators

	Name	Description
≝÷> S	BangBang	In F# code, this operator is named !!. Builds a list of Qubits from a wide variety of

			possible inputs.
	(<u>∕−</u> Ξ+) S	BangLess	In F# code, this operator is named !<. Gets the gate definition from a gate function. This is usually used to discover the "parent" gate.
	(/ =+) S	GreaterBangLessa	In F# code, this operator is named >!<. Applies a function to each qubit in a list, passing a parameter to the function as well as each Qubit. The parameter values may be a list of a single value which is then passed to each invocation.
	<u>(∕</u> Ξ+) S	GreaterLess	In F# code, this operator is named ><. Applies a function to each qubit in a list of Qubits.
	Тор		
4	See Als	0	
	Reference Microsoft.Research.Liquid Namespace		

Operations Methods

The Operations type exposes the following members.

▲ Methods

	Name	Description
= 0 S	AD	Amplitude Damping Channel using two Kraus operators (always put the most probable one first)
= 0 S	Adj	Performs the adjoint of the parent gate. This only works if the parent gate is a unitary gate.
= ∲	BC	Performs a gate under classical control. The parent gate will be executed if the control qubit has a measured value of One.
=\$ S	BCany	Performs a gate under classical control. This gate allows an arbitrary condition to be specified. The parent gate will be executed if the condition function evaluates to true.
= ≬ S	CCgate	Performs a parent gate under two- qubit quantum control. The parent must be a unitary gate.
= 0 S	CCNOT	Performs a Toffoli or Controlled- Controlled-NOT gate
= ≬ S	Cgate	Performs a parent gate under

		quantum control. The parent must be a unitary gate.
≓∳ S	CgateNC	Performs a parent gate under quantum control. The parent must be a unitary gate. The resulting gate is not cached.
= 0 S	CNOT	Performs a quantum-controlled NOT gate
= 0 S	CZ	Performs a quantum-controlled Pauli Z gate.
= ≬ S	DP	Depolarizing channel using four Kraus operators (always put the most probable one first)
≡∳ S	н	Performs a Hadamard gate.
≈\$ S	I.	Performs a Pauli I (identify) gate.
≓ ∳ S	JM	Performs a joint parity measurement in the given basis on a list of qubits. The result of the measurement is stored in the Ket's symbol table. The state vector is collapsed according to the result of the measurement.
: ♥ S	JMx	Performs a joint parity measurement in the X basis on a list of qubits. The result of the measurement is stored in the Ket's symbol table. The state vector is collapsed according to the result of the measurement.
=0 S	JMz	Performs a joint parity measurement in the computational basis on a list

		of qubits. The result of the measurement is stored in the Ket's symbol table. The state vector is collapsed according to the result of the measurement.
≕∳ S	Label	Adds a label to a circuit drawing. The label will appear above the line representing the qubit, in the center of the current column. The label will take up no logical space; it will not fill the current cell.
= \$ S	LabelC	Adds a label to a circuit drawing. The label be centered in and will fill the current cell.
≕∳ S	LabelCD	Adds a framed label to a circuit drawing. The label be centered in and will fill the current cell. The label will have a box drawn tightly around it.
≕∳ S	LabelD	Adds a label to a circuit drawing. The label will appear below the line representing the qubit, in the center of the current column. The label will take up no logical space; it will not fill the current cell.
≓Ŷ S	LabelL	Adds a label to a circuit drawing. The label will appear centered on the line representing the qubit, on the left side of the current column. The label will take up no logical space; it will not fill the current cell.

≕∳ S	LabelR	Adds a label to a circuit drawing. The label will appear centered on the line representing the qubit, on the right side of the current column. The label will take up no logical space; it will not fill the current cell.
=ŵ S	LabelRaw	Adds a raw label to a circuit drawing. The provided string should contain the Q-Circuit commands used to draw the label. Note that labels drawn using this gate will only appear when rendered to "qc" format.
= 0 S	LabelU	Adds a label to a circuit drawing. The label will appear above the line representing the qubit, in the center of the current column. The label will take up no logical space; it will not fill the current cell.
≡≬ S	LoadCache	Preload the cache with basic gates: Paulis, measurement, Hadamard, phase, CNOT, Toffoli, swap, and T.
=\$ S	Μ	Performs a measurement of a single qubit in the computational basis. The result of the measurement is stored in the measured qubit; see Qubit.Bit . The state vector is collapsed according to the result of the measurement. The measured qubit becomes "classical" and must be reset before any further quantum gates are performed on it.
≓ ≬ S	Native	Performs any desired native

		operations at this point in the circuit.
= ≬ S	NativeDbg	Performs any desired native operations at this point in the circuit. This version doesn't appear in a circuit drawing.
≓∳ S	PC	Performs a gate based on a classical condition. The parent gate is executed if the test function returns true. Typically this condition is based on the results of one or more joint parity measurements.
≡ ≬ S	R	Performs a 2pi/2^K rotation gate gate.
≓ ≬ S	Reset	Resets a qubit to a specified initial state after it has been measured. This allows quantum gates to be performed against the collapsed qubit. See the M gate.
=∳ S	Restore	Resets a qubit after it has been measured. This allows quantum gates to be performed against the collapsed qubit. The initial state of the qubit will be its last measured state. See the M gate.
≓ ≬ S	S	Performs a phase gate.
≓ ≬ S	SWAP	Performs a swap gate, exchanging the quantum states of two qubits.
≓ ≬ S	Т	Performs a pi/8 gate.
= ≬ S	T_BC	Performs a transverse classically-

¢ S	Transverse	Expands a parent gate to a transverse version.
© S	Х	Performs a Pauli X gate.
© S	Υ	Performs a Pauli Y gate.
© S	Z	Performs a Pauli Z gate.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsAD Method

Amplitude Damping Channel using two Kraus operators (always put the most probable one first)

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору _

```
static member AD :
    prob : float *
    qs : FSharpList<Qubit> -> unit
```

Parameters

prob

Type: SystemDouble Probability of damping

qs

Type: Microsoft.FSharp.CollectionsFSharpListQubit The list of qubits that the Kraus operators touch

▲ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsAdj Method

Performs the adjoint of the parent gate. This only works if the parent gate is a unitary gate.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

f

Type: Microsoft.FSharp.CoreFSharpFuncFSharpListQubit, Unit

The gate to take the adjoint of

qs

Type: **Microsoft.FSharp.CollectionsFSharpListQubit** A list of gubits which are passed to the adjoint.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsBC Method

Performs a gate under classical control. The parent gate will be executed if the control qubit has a measured value of One.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# Copy
static member BC :
    f : FSharpFunc<FSharpList<Qubit>, Unit> '
    qs : FSharpList<Qubit> -> unit
```

Parameters

f

Type: Microsoft.FSharp.CoreFSharpFuncFSharpListQubit, Unit

The parent gate to control

qs

Type: Microsoft.FSharp.CollectionsFSharpListQubit

A list of qubits. The head qubit is the control qubit; its measured state determines whether or not the parent gate is executed. The tail of the list is passed to the parent gate if it is executed.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsBCany Method

Performs a gate under classical control. This gate allows an arbitrary condition to be specified. The parent gate will be executed if the condition function evaluates to true.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

cnt

Type: SystemInt32

The count of binary control bits. These bits will not be passed to the parent gate.

tst

Type: Microsoft.FSharp.CoreFSharpFuncFSharpListQubit, Boolean

The condition function that controls execution of the parent gate.

f

Type: Microsoft.FSharp.CoreFSharpFuncFSharpListQubit, Unit

The parent gate to control

qs

Type: **Microsoft.FSharp.CollectionsFSharpListQubit** A list of qubits. The initial *cnt* qubits are provided to the condition function, and the remainder are passed to the parent gate if it is executed.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace OperationsBC(FSharpFuncFSharpListQubit, Unit, FSharpListQubit)

OperationsCCgate Method

Performs a parent gate under two-qubit quantum control. The parent must be a unitary gate.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

f

Type: Microsoft.FSharp.CoreFSharpFuncFSharpListQubit, Unit

The gate to control.

qs

Type: Microsoft.FSharp.CollectionsFSharpListQubit

The first two qubits are the control, and the remainder are passed to the parent gate.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsCCNOT Method

Performs a Toffoli or Controlled-Controlled-NOT gate

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The first two qubits in the list are the control qubits, and the third qubit is the target,

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsCgate Method

Performs a parent gate under quantum control. The parent must be a unitary gate.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# Copy
static member Cgate :
    f : FSharpFunc<FSharpList<Qubit>, Unit> '
    qs : FSharpList<Qubit> -> unit
```

Parameters

f

Type: Microsoft.FSharp.CoreFSharpFuncFSharpListQubit, Unit

The gate to control.

qs

Type: Microsoft.FSharp.CollectionsFSharpListQubit

The first qubit is the control qubit, and the remainder are passed to the parent gate.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsCgateNC Method

Performs a parent gate under quantum control. The parent must be a unitary gate. The resulting gate is not cached.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

f

Type: Microsoft.FSharp.CoreFSharpFuncFSharpListQubit, Unit

The gate to control.

qs

Type: Microsoft.FSharp.CollectionsFSharpListQubit

The first qubit is the control qubit, and the remainder are passed to the parent gate.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsCNOT Method

Performs a quantum-controlled NOT gate

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

qs

Type: **Microsoft.FSharp.CollectionsFSharpListQubit** The head qubit is the control qubit, the second qubit is the target.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsCZ Method

Performs a quantum-controlled Pauli Z gate.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

qs

Type: **Microsoft.FSharp.CollectionsFSharpListQubit** The head qubit is the control qubit, the second qubit is the target.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsDP Method

Depolarizing channel using four Kraus operators (always put the most probable one first)

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору _

```
static member DP :
    prob : float *
    qs : FSharpList<Qubit> -> unit
```

Parameters

prob

Type: SystemDouble Probability of depolarizing in X,Y or Z

qs

Type: Microsoft.FSharp.CollectionsFSharpListQubit The list of qubits that the Kraus operators touch

▲ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace
OperationsH Method

Performs a Hadamard gate.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The head qubit of this list is operated on.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

Operationsl Method

Performs a Pauli I (identify) gate.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The head qubit of this list is operated on.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsJM Method

Performs a joint parity measurement in the given basis on a list of qubits. The result of the measurement is stored in the Ket's symbol table. The state vector is collapsed according to the result of the measurement.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

```
static member JM :
    tag : string *
    basis : string *
    qs : FSharpList<Qubit> -> unit
```

Parameters

tag

Type: SystemString

If not empty, the symbol name to store the measurement result under. See **Ket.Symbol**.

basis

Type: SystemString

The basis to measure in. This may be a string of any length, including empty, made up of X, Y, and Z characters. Each character is used to specify the basis for the corresponding qubit being measured. If there are more qubits than bases specified, then the last basis character is repeated. If this is an empty string, then it is treated as "Z", which performs a joint measurement of all qubits in the computational basis.

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The list of qubits to jointly measure.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsJMx Method

Performs a joint parity measurement in the X basis on a list of qubits. The result of the measurement is stored in the Ket's symbol table. The state vector is collapsed according to the result of the measurement.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# _____Copy ____Copy ___COPY ____COPY ___COPY __COPY ___COPY ___COPY ___COPY __COPY __COPY __COPY ___COPY
```

Parameters

tag

Type: SystemString

If not empty, the symbol name to store the measurement result under. See **Ket.Symbol**.

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The list of qubits to jointly measure.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsJMz Method

Performs a joint parity measurement in the computational basis on a list of qubits. The result of the measurement is stored in the Ket's symbol table. The state vector is collapsed according to the result of the measurement.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

```
static member JMz :
    tag : string *
    qs : FSharpList<Qubit> -> unit
```

Parameters

tag

Type: SystemString

If not empty, the symbol name to store the measurement result under. See **Ket.Symbol**.

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The list of qubits to jointly measure.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsLabel Method

Adds a label to a circuit drawing. The label will appear above the line representing the qubit, in the center of the current column. The label will take up no logical space; it will not fill the current cell.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# _____Copy
static member Label :
    nam : string *
    qs : FSharpList<Qubit> -> unit
```

Parameters

nam

Type: SystemString The string to use as a label.

qs

Type: **Microsoft.FSharp.CollectionsFSharpListQubit** A list of qubits, the first of which determines the vertical positioning of the label.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsLabelC Method

Adds a label to a circuit drawing. The label be centered in and will fill the current cell.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

```
static member LabelC :
    nam : string *
    qs : FSharpList<Qubit> -> unit
```

Parameters

nam

Type: SystemString

The string to use as a label.

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit A list of qubits, the first of which determines the vertical positioning of the label.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsLabelCD Method

Adds a framed label to a circuit drawing. The label be centered in and will fill the current cell. The label will have a box drawn tightly around it.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F#
static member LabelCD :
    nam : string *
    qs : FSharpList<Qubit> -> unit
```

Copy

Parameters

nam

Type: SystemString

The string to use as a label.

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit A list of qubits, the first of which determines the vertical positioning of the label.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsLabelD Method

Adds a label to a circuit drawing. The label will appear below the line representing the qubit, in the center of the current column. The label will take up no logical space; it will not fill the current cell.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F#
Copy
static member LabelD :
    nam : string *
    qs : FSharpList<Qubit> -> unit
```

Parameters

nam

Type: SystemString The string to use as a label.

qs

Type: **Microsoft.FSharp.CollectionsFSharpListQubit** A list of qubits, the first of which determines the vertical positioning of the label.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsLabelL Method

Adds a label to a circuit drawing. The label will appear centered on the line representing the qubit, on the left side of the current column. The label will take up no logical space; it will not fill the current cell.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# Copy
static member LabelL :
    nam : string *
    qs : FSharpList<Qubit> -> unit
```

Parameters

nam

Type: SystemString The string to use as a label.

qs

Type: **Microsoft.FSharp.CollectionsFSharpListQubit** A list of qubits, the first of which determines the vertical positioning of the label.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsLabelR Method

Adds a label to a circuit drawing. The label will appear centered on the line representing the qubit, on the right side of the current column. The label will take up no logical space; it will not fill the current cell.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# Copy
static member LabelR :
    nam : string *
    qs : FSharpList<Qubit> -> unit
```

Parameters

nam

Type: SystemString The string to use as a label.

qs

Type: **Microsoft.FSharp.CollectionsFSharpListQubit** A list of qubits, the first of which determines the vertical positioning of the label.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsLabelRaw Method

Adds a raw label to a circuit drawing. The provided string should contain the Q-Circuit commands used to draw the label. Note that labels drawn using this gate will only appear when rendered to "qc" format.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

```
static member LabelRaw :
    cmd : string *
    qs : FSharpList<Qubit> -> unit
```

Parameters

cmd

Type: SystemString

The string to use to draw the label.

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit A list of qubits, the first of which determines the vertical positioning of the label.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsLabelU Method

Adds a label to a circuit drawing. The label will appear above the line representing the qubit, in the center of the current column. The label will take up no logical space; it will not fill the current cell.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F#
Copy
static member LabelU :
    nam : string *
    gs : FSharpList<Qubit> -> unit
```

Parameters

nam

Type: SystemString The string to use as a label.

qs

Type: **Microsoft.FSharp.CollectionsFSharpListQubit** A list of qubits, the first of which determines the vertical positioning of the label.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsLoadCache Method

Preload the cache with basic gates: Paulis, measurement, Hadamard, phase, CNOT, Toffoli, swap, and T.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)



OperationsM Method

Performs a measurement of a single qubit in the computational basis. The result of the measurement is stored in the measured qubit; see **Qubit.Bit**. The state vector is collapsed according to the result of the measurement. The measured qubit becomes "classical" and must be reset before any further quantum gates are performed on it.

Namespace: Microsoft.Research.Liquid

Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

```
static member M :
    qs : FSharpList<Qubit> -> unit
```

Parameters

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit A list of qubits; the head of the list is measured.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsNative Method

Performs any desired native operations at this point in the circuit.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

f

Type: Microsoft.FSharp.CoreFSharpFuncFSharpListQubit, Unit

The native function to call.

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The list of qubits to pass to the native operation.

⊿ See Also

Reference

Operations Class Microsoft.Research.Liquid Namespace OperationsNativeDbg(FSharpFuncFSharpListQubit, Unit, FSharpListQubit)

OperationsNativeDbg Method

Performs any desired native operations at this point in the circuit. This version doesn't appear in a circuit drawing.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

f

Type: Microsoft.FSharp.CoreFSharpFuncFSharpListQubit, Unit

The native function to call.

qs

```
Type: Microsoft.FSharp.CollectionsFSharpListQubit
```

The list of qubits to pass to the native operation.

J See Also

Reference

Operations Class Microsoft.Research.Liquid Namespace OperationsNative(FSharpFuncFSharpListQubit, Unit, FSharpListQubit)

OperationsPC Method

Performs a gate based on a classical condition. The parent gate is executed if the test function returns true. Typically this condition is based on the results of one or more joint parity measurements.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax F# Copy static member PC : *lbl* : string tst : FSharpFunc<FSharpList<Qubit>, bool> f : FSharpFunc<FSharpList<Qubit>, Unit> ' gs : FSharpList<Qubit> -> unit Þ Parameters lbl Type: SystemString A label for drawing. tst Type: Microsoft.FSharp.CoreFSharpFuncFSharpListQubit, Boolean The test function used to control execution of the parent gate. f Type: Microsoft.FSharp.CoreFSharpFuncFSharpListQubit, Unit The parent gate to control qs Type: Microsoft.FSharp.CollectionsFSharpListQubit

A list of qubits which will be passed to the parent gate if it is executed.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace Ket.Symbol(String, Int32)

OperationsR Method

Performs a 2pi/2^K rotation gate gate.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

k

Type: SystemInt32 The rotation parameter

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The head qubit of this list is operated on.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsReset Method

Resets a qubit to a specified initial state after it has been measured. This allows quantum gates to be performed against the collapsed qubit. See the M gate.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F#
static member Reset :
    b : Bit *
    qs : FSharpList<Qubit> -> unit
```

Copy

Parameters

b

Type: Microsoft.Research.LiquidBit

The initial state of the reset qubit, either Zero or One.

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit A list of qubits; the head of the list is reset.

▲ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsRestore Method

Resets a qubit after it has been measured. This allows quantum gates to be performed against the collapsed qubit. The initial state of the qubit will be its last measured state. See the M gate.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# _____Copy ____
static member Restore :
    gs : FSharpList<Qubit> -> unit
```

Parameters

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit A list of qubits; the head of the list is reset.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsS Method

Performs a phase gate.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The head qubit of this list is operated on.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsSWAP Method

Performs a swap gate, exchanging the quantum states of two qubits.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The states of the first two qubits in the list are swapped.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsT Method

Performs a pi/8 gate.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The head qubit of this list is operated on.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsT_BC Method

Performs a transverse classically-controlled gate.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

decode

Type: Microsoft.FSharp.CoreFSharpFuncFSharpListQubit, TupleBit, Int32

The code-specific decode function that takes a list of measured qubits and returns a logical value (One or Zero) and the Hamming distance from the current state to a valid code state.

f

Type: Microsoft.FSharp.CoreFSharpFuncFSharpListQubit, Unit

The (transverse) parent gate to control

qs

Type: Microsoft.FSharp.CollectionsFSharpListQubit

The first codeSize qubits in the list are passed to the *decode* function to compute a logical Zero or One state. If the decoded state is One, then the remaining qubits are passed to the parent gate.

⊿ See Also

Reference

Operations Class Microsoft.Research.Liquid Namespace

OperationsTransverse Method

Expands a parent gate to a transverse version.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

cCnt

Type: SystemInt32

The total number of qubits in code

f

Type: Microsoft.FSharp.CoreFSharpFuncFSharpListQubit, Unit

The parent gate to expand

qs

Type: Microsoft.FSharp.CollectionsFSharpListQubit The qubits that final gate will operate on

Remarks

If the parent gate operates on a single qubit, then it is applied to each qubit in the code. If the parent gate applies to two qubits, then it is applied to pairs of qubits selected from the first and second half of the list. For example, if the code contained 3 qubits, so that the qubits to operate on would contain 6 qubits, then the gate would be applied to pairs 0 and 3, 1 and 4, and 2 and 5. It is an error if the parent gate takes more than two qubits.

The transverse gate's name is the parent's name with a "_T" suffix.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsX Method

Performs a Pauli X gate.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The head qubit of this list is operated on.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsY Method

Performs a Pauli Y gate.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The head qubit of this list is operated on.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace
OperationsZ Method

Performs a Pauli Z gate.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The head qubit of this list is operated on.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

Operations Operators

The Operations type exposes the following members.

▲ Operators

	Name	Description
<u>(~-</u> =+) S	BangBang	In F# code, this operator is named !!. Builds a list of Qubits from a wide variety of possible inputs.
<u>(∕</u> =+) S	BangLess	In F# code, this operator is named !<. Gets the gate definition from a gate function. This is usually used to discover the "parent" gate.
≝÷ S	GreaterBangLessa	In F# code, this operator is named >!<. Applies a function to each qubit in a list, passing a parameter to the function as well as each Qubit. The parameter values may be a list of a single value which is then passed to each invocation.
⊻= =+) S	GreaterLess	In F# code, this operator is named ><. Applies a function to each qubit in a list of Qubits.

Тор

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsBangBang Operator

In F# code, this operator is named !!. Builds a list of Qubits from a wide variety of possible inputs.

_ Copy _

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

F# does not support this operator.

Parameters

qs

Type: SystemObject The value to interpret as a list of Qubits.

Return Value Type: **FSharpListQubit** A list of Qubits

Remarks

The inputs that are interprested by this operator are:

- A Ket, which is interpreted as the complete list of Qubits in the Ket.
- A single Qubit, which is interpreted as a single-element list.
- A two-tuple of Qubits, which is interpreted as a two-element list.
- A three-tuple of Qubits, which is interpreted as a threeelement list.
- A list of Qubits, which is returned directly.

- A two-tuple of lists of Qubits, which are concatenated into a single list.
- A three-tuple of lists of Qubits, which are concatenated into a single list.
- A list of lists of Qubits, which are concatenated into a single list.
- A two-tuple of a list of Qubits and a single Qubit, which are concatenated into a single list.
- A two-tuple of a single Qubit and a list of Qubits, which are concatenated into a single list.
- A two-tuple of a list of Qubits and an integer, which is interpreted as the single-element list containing the item in the list indexed by the integer.
- A three-tuple of a list of Qubits and two integers, which is interpreted as the two-element list containing the items in the list indexed by the two integers.
- A four-tuple of a list of Qubits and three integers, which is interpreted as the three-element list containing the item in the list indexed by the three integers.
- A two-tuple of a list of Qubits and a list of integers, which is interpreted as the list containing the items in the Qubit list indexed by the elements in the integer list.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsBangLess Operator

In F# code, this operator is named !<. Gets the gate definition from a gate function. This is usually used to discover the "parent" gate.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

_ Copy _

F# does not support this operator.

Parameters

f

Type: Microsoft.FSharp.CoreFSharpFuncFSharpListQubit, Unit

The gate function we want the Gate for.

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit A list of Qubits that can be used to find the gate.

Return Value

Type: Gate

The Gate corresponding to the gate function. Note that if f is not a gate function, this function will raise an exception.

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsGreaterBangLessa Operator

In F# code, this operator is named >!<. Applies a function to each qubit in a list, passing a parameter to the function as well as each Qubit. The parameter values may be a list of a single value which is then passed to each invocation.

Namespace: Microsoft.Research.Liquid

Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Copy

F# does not support this operator.

Parameters

f

Type: Microsoft.FSharp.CoreFSharpFunca, FSharpFuncFSharpListQubit, Unit

The function to call. The extra parameter must be the first argument to the function.

args

Type: SystemObject

Either a list of arguments, one per Qubit, or a single value that is passed with each Qubit.

qs

Type: Microsoft.FSharp.CollectionsFSharpListQubit The list of qubits to iterate over.

Type Parameters

а

⊿ See Also

Reference Operations Class Microsoft.Research.Liquid Namespace

OperationsGreaterLess Operator

In F# code, this operator is named ><. Applies a function to each qubit in a list of Qubits.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

___ Сору __

F# does not support this operator.

Parameters

f

Type: Microsoft.FSharp.CoreFSharpFuncFSharpListQubit, Unit

The gate function to call.

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The list of qubits to iterate over.

⊿ See Also

Reference

Operations Class Microsoft.Research.Liquid Namespace

QECC Class

Base class for quantum error correcting codes.

Inheritance Hierarchy

```
SystemObject Microsoft.Research.LiquidQECC
Microsoft.Research.LiquidSteane7
```

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



The QECC type exposes the following members.

Constructors

	Name	Description
= \$	QECC	Initializes a Quantum Error Correcting Circuit (QECC). This base constructor must be called by all derived constructors.

Тор

Properties

	Name	Description
*	Circuit	The error-correcting circuit built by Compile. The circuit will be built now if it hasn't already been.
	Ket	The state vector for the compiled code.

Тор

▲ Methods

	Name	Description
≓∳	Compile	Compiles the target circuit into an error-correcting version.
≡∳	Decode	Decodes a set of measured physical qubits to get the measured value for a logical qubit.
=♥	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
≞ ©	GetHashCode	Serves as the default hash function. (Inherited from Object.)
≕ ©	GetMeasured	Gets the measured values of the physical qubits that make up a logical qubit and returns them combined into a single integer, one bit per qubit.
= Q	GetType	Gets the Type of the current

		instance. (Inherited from Object.)
-= Q	Inject	Injects dephasing errors with the given probability into the error-correcting circuit.
-= Q	Log2Phys	Gets the physical qubits that make up a logical qubit.
-=•	Prep	A gate function that prepares a logical 0> qubit. This gets compiled into the error-correcting circuit by the Compile method.
a∳	Replace	Gets a replacement physical gate for an input logical gate. The replacement may wrap a full Circuit.
.≡♥	Syndrome	A gate function that measures the physical qubits for a single logical qubit and applies any necessary corrections. This gets compiled into the error-correcting circuit by the Compile method.
-≓∳	ToString	Returns a string that represents the current object. (Inherited from Object.)

Тор

⊿ See Also

Reference Microsoft.Research.Liquid Namespace

QECC Constructor

Initializes a Quantum Error Correcting Circuit (QECC). This base constructor must be called by all derived constructors.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

new : aCnt : int * cCnt : int * tgt : Circuit -> QECC

Parameters

aCnt

Type: SystemInt32

The number of ancilla qubits. By convention ancillae come first in the state vector, before data and syndrome qubits. Ancilla qubits are shared across all logical qubits and are for use during decoding, measurement, and other operations; they are not syndrome qubits.

Copy

cCnt

Type: SystemInt32

The code size; that is, the number of physical qubits (data and syndrome) per logical qubit.

tgt

Type: Microsoft.Research.LiquidCircuit

The target Circuit to build an error-correcting circuit for.

⊿ See Also

Reference

QECC Class Microsoft.Research.Liquid Namespace

QECC Properties

The QECC type exposes the following members.

▲ Properties

		Name	Description	
	*	Circuit	The error-correcting circuit built by Compile. The circuit will be built now if i hasn't already been.	
		Ket	The state vector for the compiled code.	
	Тор			
4	See Also			
	Reference QECC Class Microsoft.Research.Liquid Namespace			

QECCCircuit Property

The error-correcting circuit built by Compile. The circuit will be built now if it hasn't already been.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

___ Сору _

member Circuit : Circuit with get

Property Value Type: Circuit

⊿ See Also

Reference QECC Class Microsoft.Research.Liquid Namespace

QECCKet Property

The state vector for the compiled code.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



QECC Class Microsoft.Research.Liquid Namespace

QECC Methods

The **QECC** type exposes the following members.

▲ Methods

	Name	Description
≡ Q	Compile	Compiles the target circuit into an error-correcting version.
=∳	Decode	Decodes a set of measured physical qubits to get the measured value for a logical qubit.
=∳	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
=♥	GetHashCode	Serves as the default hash function. (Inherited from Object.)
=♥	GetMeasured	Gets the measured values of the physical qubits that make up a logical qubit and returns them combined into a single integer, one bit per qubit.
=♥	GetType	Gets the Type of the current instance. (Inherited from Object.)

- =	Inject	Injects dephasing errors with the given probability into the error-correcting circuit.
-= Q	Log2Phys	Gets the physical qubits that make up a logical qubit.
-=	Prep	A gate function that prepares a logical 0> qubit. This gets compiled into the error-correcting circuit by the Compile method.
-= Q	Replace	Gets a replacement physical gate for an input logical gate. The replacement may wrap a full Circuit.
=♥	Syndrome	A gate function that measures the physical qubits for a single logical qubit and applies any necessary corrections. This gets compiled into the error-correcting circuit by the Compile method.
≣∳	ToString	Returns a string that represents the current object. (Inherited from Object.)

Тор

⊿ See Also

Reference QECC Class Microsoft.Research.Liquid Namespace

QECCCompile Method

Compiles the target circuit into an error-correcting version.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



⊿ See Also

Reference QECC Class Microsoft.Research.Liquid Namespace

QECCDecode Method

Decodes a set of measured physical qubits to get the measured value for a logical qubit.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

qs

Type: Microsoft.FSharp.CollectionsFSharpListQubit

The physical qubits to decode. They must already have been measured.

Return Value

Type: TupleBit, Int32

A tuple containing the logical measured value of the logical qubit, either Zero or One, and the Hamming distance from the physical state to the code space.

⊿ See Also

Reference QECC Class Microsoft.Research.Liquid Namespace

QECCGetMeasured Method

Gets the measured values of the physical qubits that make up a logical qubit and returns them combined into a single integer, one bit per qubit.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

```
member GetMeasured :
    gs : FSharpList<Qubit> -> int
```

Parameters

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The set of qubits to read. This should be the set of physical (data and syndrome) qubits for one logical qubit.

Copy

Return Value

Type: Int32

The total measured value of the qubits. The value of the first qubit in the list goes into the left-most (most significant) bit in the result.

⊿ See Also

Reference QECC Class Microsoft.Research.Liquid Namespace

QECCInject Method

Injects dephasing errors with the given probability into the errorcorrecting circuit.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# _____Copy
member Inject :
    prob : float -> Tuple<Circuit, FSharpList
</pre>
```

Parameters

prob

Type: SystemDouble

The probability of an error on execution of a gate. Each qubit input to the gate has this same chance of a dephasing error. If an error is injected, it will be either an X, Y, or Z error, with equal probability. Note that errors are never injected on ancilla qubits.

Return Value

Type: TupleCircuit, FSharpListInt32

A tuple containing the new Circuit with errors injected as the first item and a list of injected error counts as the second. The error count list contains the count of X, Y, and Z errors injected, in that order.

Remarks

Note that the errors are inserted as explicit gates into a new Circuit,

rather than randomly being injected on each run. That is, if the resulting circuit is executed multiple times, the same errors will be injected each time. To get a different set of random errors, this routine must be re-run and a new Circuit generated. Also note that errors are only inserted before wrapped and extended gates (Gate types Wrap and Ext). In particular, this means that errors appear before logical gates rather than before physical gates. If the original non-error correcting circuit included wrapped or extended gates, then errors may be injected before those gates as well.

▲ See Also

Reference QECC Class Microsoft.Research.Liquid Namespace

QECCLog2Phys Method

Gets the physical qubits that make up a logical qubit.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

W

Type: SystemInt32 The wire ID of the logical qubit in the original non-error correcting circuit.

Return Value Type: **FSharpListQubit** A list of the physical qubits that implement the logical qubit.

⊿ See Also

Reference QECC Class Microsoft.Research.Liquid Namespace

QECCPrep Method

A gate function that prepares a logical |0> qubit. This gets compiled into the error-correcting circuit by the Compile method.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F#
    Copy
abstract Prep :
    gs : FSharpList<Qubit> -> unit
```

Parameters

qs

Type: **Microsoft.FSharp.CollectionsFSharpListQubit** The physical qubits for the logical qubit. This contains both data and syndrome qubits, but no ancillae.

⊿ See Also

Reference QECC Class Microsoft.Research.Liquid Namespace

QECCReplace Method

Gets a replacement physical gate for an input logical gate. The replacement may wrap a full Circuit.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

g

Type: Microsoft.Research.LiquidGate The logical gate function to replace.

Return Value

Type: FSharpOptionFSharpFuncFSharpListQubit, Unit

An option holding a physical gate that implements the logical gate. If there is no physical implementation of the logical gate, then None is returned.

Remarks

The default implementation replaces the Pauli gates, phase gate, Hadamard gate, identity gate, CNOT, and measurement with transverse versions of these gates. Classically-controlled versions of these gates are replaced by transverse versions, controlled by the decoded (logical) value of the classical control.

⊿ See Also

Reference QECC Class Microsoft.Research.Liquid Namespace

QECCSyndrome Method

A gate function that measures the physical qubits for a single logical qubit and applies any necessary corrections. This gets compiled into the error-correcting circuit by the Compile method.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# _____Copy ____
abstract Syndrome :
    gs : FSharpList<Qubit> -> unit
```

Parameters

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The ancilla qubits followed by physical qubits for the logical qubit. The number of ancillae was passed to the QECC constructor.

⊿ See Also

Reference

QECC Class Microsoft.Research.Liquid Namespace

Qubit Class

Represents a quantum bit. New Qubits are created using the Ket Add methods.

▲ Inheritance Hierarchy

```
SystemObject Microsoft.Research.LiquidQubit
```

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# _____Copy ____
[<SerializableAttribute>]
type Qubit = class end
```

The Qubit type exposes the following members.

▲ Properties

Name	Description
Bit	The measured value of a qubit in the computational basis. This will be Unknown if the qubit has not been measured since it last interacted.
Entangled	Whether or not this qubit is entangled.
Id	Wire number in Ket vector
Ket	State we belong to

🕈 S	One	A Complex vector that represents the state 1> in the computational basis.
	Prob1	The probability of this qubit being 1.
**	State	The state vector for this qubit, if it is unentangled. This property will raise an exception if the qubit is entangled.
*	Туре	The type of the qubit. This is only used for noise modeling.
🕈 S	Zero	A Complex vector that represents the state 0> in the computational basis.

Тор

▲ Methods

	Name	Description
=	Dump	Dump this qubit's state.
=♥	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
=♥	GetHashCode	Serves as the default hash function. (Inherited from Object.)
=♥	GetType	Gets the Type of the current instance. (Inherited from Object.)
≡\$	ReAnimate(Bit)	Reanimates this qubit and

		sets its state vector to 0> or 1>. Reanimation means that this qubit is treated as unmeasured and eligible for quantum operations. This method will raise an exception if the qubit is unmeasured or entangled.
=∲	ReAnimate(CVec)	Reanimates this qubit and sets its state vector. Reanimation means that this qubit is treated as unmeasured and eligible for quantum operations. This method will raise an exception if the qubit is unmeasured or entangled.
≓∳	ShowMag	Creates a string representation of this Qubit. The representation shows the probabilities of measuring this Qubit in the computational 0> and 1> states.
≡\$	StateSet(Bit)	Sets the state vector for this qubit to 0> or 1>, if it is unentangled. This method will raise an exception if the qubit is entangled.
≡\$	StateSet(CVec)	Sets the state vector for this qubit, if it is unentangled. This method will raise an exception if the qubit is entangled.
\$	StateSet(Complex, Complex)	Sets the state vector for this qubit, if it is unentangled. This method will raise an exception if the qubit is entangled.
-----------	--	---
9 	StateSet(Double, Double, Double, Double)	Sets the state vector for this qubit, if it is unentangled. This method will raise an exception if the qubit is entangled.
٩	ToString	Creates a string representation of this Qubit. The representation shows the current state of the qubit if it is unentangled. (Overrides ObjectToString.)

Тор

⊿ See Also

Reference Microsoft.Research.Liquid Namespace

Qubit Properties

The Qubit type exposes the following members.

▲ Properties

	Name	Description
	Bit	The measured value of a qubit in the computational basis. This will be Unknown if the qubit has not been measured since it last interacted.
	Entangled	Whether or not this qubit is entangled.
	Id	Wire number in Ket vector
	Ket	State we belong to
🕈 s	One	A Complex vector that represents the state 1> in the computational basis.
	Prob1	The probability of this qubit being 1.
*	State	The state vector for this qubit, if it is unentangled. This property will raise an exception if the qubit is entangled.
*	Туре	The type of the qubit. This is only used for noise modeling.
📽 s	Zero	A Complex vector that represents the state 0> in the computational basis.

Тор

⊿ See Also

Reference Qubit Class Microsoft.Research.Liquid Namespace

QubitBit Property

The measured value of a qubit in the computational basis. This will be Unknown if the qubit has not been measured since it last interacted.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Reference Qubit Class Microsoft.Research.Liquid Namespace

QubitEntangled Property

Whether or not this qubit is entangled.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

member Entangled : bool with get

Property Value Type: Boolean

⊿ See Also

Reference Qubit Class Microsoft.Research.Liquid Namespace

QubitId Property

Wire number in Ket vector

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# ______ Copy _____ member Id : int with get Property Value Type: Int32 • See Also

Reference Qubit Class

Microsoft.Research.Liquid Namespace

QubitKet Property

State we belong to

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Qubit Class Microsoft.Research.Liquid Namespace

QubitOne Property

A Complex vector that represents the state $|1\rangle$ in the computational basis.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)



Property Value Type: CVec

⊿ See Also

Reference Qubit Class Microsoft.Research.Liquid Namespace

QubitProb1 Property

The probability of this qubit being 1.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

_ Copy _

member Prob1 : float with get

Property Value Type: Double

⊿ See Also

Reference Qubit Class Microsoft.Research.Liquid Namespace

QubitState Property

The state vector for this qubit, if it is unentangled. This property will raise an exception if the qubit is entangled.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

___ Copy _

▲ Syntax

F#

member State : CVec with get

Property Value Type: CVec

⊿ See Also

Reference Qubit Class Microsoft.Research.Liquid Namespace

QubitType Property

The type of the qubit. This is only used for noise modeling.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Reference Qubit Class Microsoft.Research.Liquid Namespace

QubitZero Property

A Complex vector that represents the state |0> in the computational basis.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору _

static member Zero : CVec with get

Property Value Type: CVec

⊿ See Also

Reference Qubit Class Microsoft.Research.Liquid Namespace

Qubit Methods

The Qubit type exposes the following members.

Methods

	Name	Description
≡ ⊘	Dump	Dump this qubit's state.
.	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
=0	GetHashCode	Serves as the default hash function. (Inherited from Object.)
=♥	GetType	Gets the Type of the current instance. (Inherited from Object.)
=>	ReAnimate(Bit)	Reanimates this qubit and sets its state vector to 0> or 1>. Reanimation means that this qubit is treated as unmeasured and eligible for quantum operations. This method will raise an exception if the qubit is unmeasured or entangled.
≡∳	ReAnimate(CVec)	Reanimates this qubit and sets its state vector.

		Reanimation means that this qubit is treated as unmeasured and eligible for quantum operations. This method will raise an exception if the qubit is unmeasured or entangled.
. ≕	ShowMag	Creates a string representation of this Qubit. The representation shows the probabilities of measuring this Qubit in the computational 0> and 1> states.
≡ ©́	StateSet(Bit)	Sets the state vector for this qubit to 0> or 1>, if it is unentangled. This method will raise an exception if the qubit is entangled.
≡ ©	StateSet(CVec)	Sets the state vector for this qubit, if it is unentangled. This method will raise an exception if the qubit is entangled.
≡ ©	StateSet(Complex, Complex)	Sets the state vector for this qubit, if it is unentangled. This method will raise an exception if the qubit is entangled.
≡©	StateSet(Double, Double, Double, Double)	Sets the state vector for this qubit, if it is unentangled. This method will raise an exception if the qubit is

		entangled.
≓ \$	ToString	Creates a string representation of this Qubit. The representation shows the current state of the qubit if it is unentangled. (Overrides ObjectToString.)

Тор

⊿ See Also

Reference Qubit Class Microsoft.Research.Liquid Namespace

QubitDump Method

Dump this qubit's state.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

f

Type: Microsoft.FSharp.CoreFSharpOptionFSharpFuncInt32, FSharpFuncString, Unit

The optional output function to use. The default is showLogInd. *level*

Type: **Microsoft.FSharp.CoreFSharpOptionInt32** The optional indentation level. The default is 0.

⊿ See Also

Reference

Qubit Class Microsoft.Research.Liquid Namespace

QubitReAnimate Method

Overload List

	Name	Description
.≡∳	ReAnimate(Bit)	Reanimates this qubit and sets its state vector to 0> or 1>. Reanimation means that this qubit is treated as unmeasured and eligible for quantum operations. This method will raise an exception if the qubit is unmeasured or entangled.
.≕∳	ReAnimate(CVec)	Reanimates this qubit and sets its state vector. Reanimation means that this qubit is treated as unmeasured and eligible for quantum operations. This method will raise an exception if the qubit is unmeasured or entangled.

Тор

⊿ See Also

Reference Qubit Class Microsoft.Research.Liquid Namespace

QubitReAnimate Method (Bit)

Reanimates this qubit and sets its state vector to |0> or |1>. Reanimation means that this qubit is treated as unmeasured and eligible for quantum operations. This method will raise an exception if the qubit is unmeasured or entangled.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

```
member ReAnimate :
    b : Bit -> unit
```

Parameters

b

Type: Microsoft.Research.LiquidBit The Bit value to set this qubit's state to.

⊿ See Also

Reference Qubit Class ReAnimate Overload Microsoft.Research.Liquid Namespace

QubitReAnimate Method (CVec)

Reanimates this qubit and sets its state vector. Reanimation means that this qubit is treated as unmeasured and eligible for quantum operations. This method will raise an exception if the qubit is unmeasured or entangled.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

Parameters

v

Type: Microsoft.Research.LiquidCVec The Complex vector to set this qubit's state to.

⊿ See Also

Reference Qubit Class ReAnimate Overload Microsoft.Research.Liquid Namespace

QubitShowMag Method

Creates a string representation of this Qubit. The representation shows the probabilities of measuring this Qubit in the computational |0> and |1> states.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____Copy
member ShowMag : unit -> string

Return Value Type: String The string representation

⊿ See Also

Reference Qubit Class Microsoft.Research.Liquid Namespace [M:Microsoft.Research.Liquid.Qubit.ToString()]

QubitStateSet Method

Overload List

	Name	Description
=♥	StateSet(Bit)	Sets the state vector for this qubit to 0> or 1>, if it is unentangled. This method will raise an exception if the qubit is entangled.
=♥	StateSet(CVec)	Sets the state vector for this qubit, if it is unentangled. This method will raise an exception if the qubit is entangled.
=∳	StateSet(Complex, Complex)	Sets the state vector for this qubit, if it is unentangled. This method will raise an exception if the qubit is entangled.
=∳	StateSet(Double, Double, Double, Double)	Sets the state vector for this qubit, if it is unentangled. This method will raise an exception if the qubit is entangled.

Тор

⊿ See Also

Reference Qubit Class Microsoft.Research.Liquid Namespace

QubitStateSet Method (Bit)

Sets the state vector for this qubit to |0> or |1>, if it is unentangled. This method will raise an exception if the qubit is entangled.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____Copy _____Copy _____Copy _____Copy ______Copy ______

Parameters

b

Type: Microsoft.Research.LiquidBit The Bit value to set this qubit's state to.

⊿ See Also

Reference Qubit Class StateSet Overload Microsoft.Research.Liquid Namespace

QubitStateSet Method (CVec)

Sets the state vector for this qubit, if it is unentangled. This method will raise an exception if the qubit is entangled.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

v

Type: Microsoft.Research.LiquidCVec The Complex vector to set this qubit's state to.

⊿ See Also

Reference Qubit Class StateSet Overload Microsoft.Research.Liquid Namespace

QubitStateSet Method (Complex, Complex)

Sets the state vector for this qubit, if it is unentangled. This method will raise an exception if the qubit is entangled.

Copy

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

member StateSet :

c1 : Complex *

c2 : Complex -> unit

Parameters

c1

Type: Microsoft.Research.LiquidComplex The complex |0> amplitude to set this qubit's state to.

с2

Type: Microsoft.Research.LiquidComplex The complex |1> amplitude to set this qubit's state to.

J See Also

Reference Qubit Class StateSet Overload Microsoft.Research.Liquid Namespace

QubitStateSet Method (Double, Double, Double, Double, Double, Double)

Sets the state vector for this qubit, if it is unentangled. This method will raise an exception if the qubit is entangled.

Copy

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Stat	es	Set :		
r1	:	float	*	
<i>i1</i>	:	float	*	
r2	:	float	*	
i2	:	float	->	unit
	Stat <i>r1</i> <i>i1</i> <i>r2</i> <i>i2</i>	States <i>r1</i> : <i>i1</i> : <i>r2</i> : <i>i2</i> :	<pre>StateSet : r1 : float i1 : float r2 : float i2 : float</pre>	<pre>StateSet : r1 : float * i1 : float * r2 : float * i2 : float -></pre>

Parameters

r1

Type: SystemDouble

The real part of the |0> amplitude to set this qubit's state to.

i1

Type: SystemDouble

The imaginary part of the |0> amplitude to set this qubit's state to.

r2

Type: SystemDouble

The real part of the |1> amplitude to set this qubit's state to.

i2

Type: SystemDouble

The imaginary part of the |1> amplitude to set this qubit's state

to.

⊿ See Also

Reference Qubit Class StateSet Overload Microsoft.Research.Liquid Namespace

QubitToString Method

Creates a string representation of this Qubit. The representation shows the current state of the qubit if it is unentangled.

Copy

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

abstract ToString : unit -> string
override ToString : unit -> string

Return Value Type: String The string representation

⊿ See Also

Reference Qubit Class Microsoft.Research.Liquid Namespace [M:Microsoft.Research.Liquid.Qubit.ShowMag()]

RunMode Class

Trotterization types.

▲ Inheritance Hierarchy

SystemObject Microsoft.Research.LiquidRunMode

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#	Сору			
[<sealedattribute>] [<serializableattribute>]</serializableattribute></sealedattribute>				
type RunMode =				
CLASS				
<pre>interface IEquatable<runmode></runmode></pre>				
interface IStructuralEquatable				
<pre>interface IComparable<runmode></runmode></pre>				
interface IComparable				
interface IStructuralComparable				
end				

The RunMode type exposes the following members.

Methods

	Name	Description
≓∳	Equals	Determines whether the specified object is equal to the current object.

		(Inherited from Object.)
= Q	GetHashCode	Serves as the default hash function. (Inherited from Object.)
-= Q	GetType	Gets the Type of the current instance. (Inherited from Object.)
≡♥	ToString	Returns a string that represents the current object. (Inherited from Object.)

Тор

Remarks

Possible values are:

- Trotter1: First order Trotter
- Trotter1X: First order Trotter splitting X to each side
- Trotter1R: Reverse mode (implies reversing list and negating angles)
- **Trotter1XR**: Reverse mode with X splitting (implies reversing list and negating angles)
- Trotter2: Second order Trotter (includes X splitting)
- Trotter2R: Second order reversed (includes X splitting)

⊿ See Also

Reference

Microsoft.Research.Liquid Namespace

RunMode Methods

The RunMode type exposes the following members.

Methods

	Name	Description
=♥	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
=∳	GetHashCode	Serves as the default hash function. (Inherited from Object.)
=≬	GetType	Gets the Type of the current instance. (Inherited from Object.)
=♥	ToString	Returns a string that represents the current object. (Inherited from Object.)

Тор

⊿ See Also

Reference RunMode Class Microsoft.Research.Liquid Namespace

Spin Class

Hamiltonian for spin systems, such as the Ising model or a spin glass.

Inheritance Hierarchy

```
SystemObject Microsoft.Research.LiquidHamiltonian 
Microsoft.Research.LiquidSpin
```

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



The Spin type exposes the following members.

Constructors

Name	Description
Spin(DictionaryIn Double, DictionaryTupleIn Int32, Double)	tt32, Creates a spin Hamiltonian for simple systems that only have single spin biases and two-spin couplings. The count of spins is inferred.

= \$	Spin(FSharpListSpinTerm,	Creates a spin
li li	nt32,	Hamiltonian from a
F	SharpOptionRunMode)	set of spin terms.

Тор

Properties

	Name	Description
*	currentCirc	The last (grown) circuit that was run.
	decohereModel	The decoherence model for this Hamiltonian. (Inherited from Hamiltonian.)
	Ket	Gets the Ket vector associated with this Hamiltonian (Inherited from Hamiltonian.)
	lastAnneal	The last set of annealing coefficients that were applied.
	lastRawCirc	The last (ungrown) circuit that was run.
	runMode	The run mode. Note that setting this property will force a new circuit to be generated.
	time	The current simulation time. Note that setting this property will force a new circuit to be generated.
	trotterN	The Trotter number. Note that setting this property will force a
new circuit to be generated.

Тор

▲ Methods

	Name	Descrip
	Copy	Returns instance Spin with same Hamiltor paramet this insta Simulation paramet such as time and number copied.
-= Q	EnergyExpectation	Finds the expectat value of Hamiltor That is, (state vec <i>psi</i> >, thi method compute < <i>psi</i> H <i>p</i>
-= Q	Equals	Determiı whether specifiec is equal

		current c (Inherite Object.)
≋ ∳ S	Ferro	Test adia evolution ferro-ma chain
= \$	GetHashCode	Serves a default h function. (Inherite Object.)
= ()	GetType	Gets the of the cu instance (Inherite Object.)
≡()	Peek	Peeks a current Hamiltor matrix. T based o most rec grown ci
≡∲	Prep	Prepares qubit sta run. All c are rese initializes ground s
=	Run	Runs the simulatic

		is a high optimize impleme
≡♥	Step	Performs single tir of simula
= 0 S	Test(String, Int32, Int32, FSharpListTupleInt32, Double, Int32, Spin, FSharpOptionBoolean, FSharpOptionFSharpListTupleFSharpFuncQubit, Unit, Double)	Execute of simula runs for Hamiltor
= 0 S	Test(String, Int32, Int32, DictionaryInt32, Double, DictionaryTupleInt32, Int32, Double, FSharpListTupleInt32, Double, Double, Int32, FSharpOptionBoolean, FSharpOptionFSharpListTupleFSharpFuncQubit, Unit, Double)	Execute of simula runs for Hamiltor
=♥	ToString	Returns that repr the curre object. (Inherite Object.)

⊿ See Also

Reference Microsoft.Research.Liquid Namespace

Spin Constructor

Overload List

	Name	Description
≕©	Spin(DictionaryInt32, Double, DictionaryTupleInt32, Int32, Double)	Creates a spin Hamiltonian for simple systems that only have single spin biases and two-spin couplings. The count of spins is inferred.
÷Ŷ	Spin(FSharpListSpinTerm, Int32, FSharpOptionRunMode)	Creates a spin Hamiltonian from a set of spin terms.
Тор		
See A	Also	

Reference Spin Class Microsoft.Research.Liquid Namespace

Spin Constructor (DictionaryInt32, Double, DictionaryTupleInt32, Int32, Double)

Creates a spin Hamiltonian for simple systems that only have single spin biases and two-spin couplings. The count of spins is inferred.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

hs

Type: System.Collections.GenericDictionaryInt32, Double A Dictionary that maps a qubit id to that qubit's bias strength (Z term coefficient).

Js

Type: System.Collections.GenericDictionaryTupleInt32, Int32, Double

A Dictionary that maps a pair of qubit id's to the pair's coupling strength (ZZ coefficient).

⊿ See Also

Reference

Spin Class Spin Overload Microsoft.Research.Liquid Namespace

Spin Constructor (FSharpListSpinTerm, Int32, FSharpOptionRunMode)

Creates a spin Hamiltonian from a set of spin terms.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# _____Copy ____
new :
    ____spinTerms : FSharpList<SpinTerm> *
    ___numSpins : int *
    __runMode : FSharpOption<RunMode> -> Spin
```

Parameters

_spinTerms

Type: **Microsoft.FSharp.CollectionsFSharpList**SpinTerm The list of SpinTerms that together make up the spin Hamiltonian.

_numSpins

Type: SystemInt32

The number of spins in the system.

_runMode

Type: Microsoft.FSharp.CoreFSharpOptionRunMode

The Trotterization type to use. This must be one of Trotter1, Trotter1X, or Trotter2.

⊿ See Also

Reference

Spin Class Spin Overload Microsoft.Research.Liquid Namespace

Spin Properties

The Spin type exposes the following members.

▲ Properties

	Name	Description
*	currentCirc	The last (grown) circuit that was run.
*	decohereModel	The decoherence model for this Hamiltonian. (Inherited from Hamiltonian.)
*	Ket	Gets the Ket vector associated with this Hamiltonian (Inherited from Hamiltonian.)
*	lastAnneal	The last set of annealing coefficients that were applied.
*	lastRawCirc	The last (ungrown) circuit that was run.
*	runMode	The run mode. Note that setting this property will force a new circuit to be generated.
	time	The current simulation time. Note that setting this property will force a new circuit to be generated.
	trotterN	The Trotter number. Note that

setting this property will force a new circuit to be generated.

Тор



Reference Spin Class Microsoft.Research.Liquid Namespace

SpincurrentCirc Property

The last (grown) circuit that was run.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

member currentCirc : Circuit with get

Property Value Type: Circuit

⊿ See Also

Reference Spin Class Microsoft.Research.Liquid Namespace

SpinlastAnneal Property

The last set of annealing coefficients that were applied.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#
 copy
member lastAnneal : float[] with get
Property Value
Type: Double
A See Also

Reference Spin Class Microsoft.Research.Liquid Namespace

SpinlastRawCirc Property

The last (ungrown) circuit that was run.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

member lastRawCirc : Circuit with get

Property Value Type: Circuit

⊿ See Also

Reference Spin Class Microsoft.Research.Liquid Namespace

SpinrunMode Property

The run mode. Note that setting this property will force a new circuit to be generated.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____Copy
member runMode : RunMode with get, set

Property Value Type: RunMode

Remarks

Possibilities are Trotter1 (fully first-order); Trotter1X (does X/2 on both sides, but fully first-order in other terms); and Trotter2 (fully second-order). The initial default value for new Hamiltonians is Trotter1.

⊿ See Also

Reference Spin Class Microsoft.Research.Liquid Namespace

Spintime Property

The current simulation time. Note that setting this property will force a new circuit to be generated.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax F# _____Copy __ member time : int with get, set Property Value Type: Int32

⊿ See Also

Reference Spin Class Microsoft.Research.Liquid Namespace

SpintrotterN Property

The Trotter number. Note that setting this property will force a new circuit to be generated.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

_ Сору _

member trotterN : int with get, set

Property Value Type: Int32

⊿ See Also

Reference Spin Class Microsoft.Research.Liquid Namespace

Spin Methods

The Spin type exposes the following members.

Methods

	Name	Descrip
	Copy	Returns instance Spin witl same Hamiltor paramet this insta Simulatio paramet such as time anc number copied.
=♥	EnergyExpectation	Finds the expectal value of Hamiltor That is, e state vec [<i>psi</i> >, thi method compute < <i>psi</i> H <i>p</i>
≡∲	Equals	Determiı

		whether specifiec is equal current c (Inherite Object.)
≋∳ S	Ferro	Test adia evolutior ferro-ma chain
.≓ ∲	GetHashCode	Serves a default h function. (Inherite Object.)
æ	GetType	Gets the of the cu instance (Inherite Object.)
≡0	Peek	Peeks a current Hamiltor matrix. T based o most rec grown ci
≡∲	Prep	Prepares qubit sta run. All c are rese initializes ground s

≓\$	Run	Runs the simulation is a high optimize impleme
≡Ŵ	Step	Performs single tir of simula
≓\$ S	Test(String, Int32, Int32, FSharpListTupleInt32, Double, Int32, Spin, FSharpOptionBoolean, FSharpOptionFSharpListTupleFSharpFuncQubit, Unit, Double)	Execute: of simula runs for Hamiltor
≡∳ S	Test(String, Int32, Int32, DictionaryInt32, Double, DictionaryTupleInt32, Int32, Double, FSharpListTupleInt32, Double, Double, Int32, FSharpOptionBoolean, FSharpOptionFSharpListTupleFSharpFuncQubit, Unit, Double)	Execute: of simula runs for Hamiltor
≡∳	ToString	Returns that repr the curre object. (Inherite Object.)

Reference Spin Class Microsoft.Research.Liquid Namespace

SpinCopy Method

Returns an instance of Spin with the same Hamiltonian parameters as this instance. Simulation parameters such as run time and Trotter number are **not** copied.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Return Value Type: Spin Copy of this instance

⊿ See Also

Reference Spin Class Microsoft.Research.Liquid Namespace

SpinEnergyExpectation Method

Finds the expectation value of the Hamiltonian. That is, given a state vector |psi>, this method computes $<psi|\mathbf{H}|psi>$.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

stdev

Type: SystemBoolean

Whether or not to evaluate the standard deviation as well. *anneal*

Type: Microsoft.FSharp.CoreFSharpOptionDouble

An optional array of annealing values to use. The default is to use the most recent annealing values.

qubits

Type: **Microsoft.FSharp.CoreFSharpOptionFSharpListQubit** An optional state vector to take the expectation value against. The default is to use this.Ket.Qubits; that is, the qubits from the current Ket vector.

Return Value Type: TupleDouble, Double A tuple of the expectation value and standard deviation. The standard deviation will be 0.0 if it was not computed.

⊿ See Also

Reference Spin Class Microsoft.Research.Liquid Namespace

SpinFerro Method

Test adiabatic evolution with a ferro-magnetic chain

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Type: SystemDouble -1,0,1 = right most down, random, up J

Type: SystemDouble

coupling: 1=ferro -1=anti 0=none

gammalambda

Type: Microsoft.FSharp.CoreFSharpOptionFSharpListTupleInt Double, Double

List of (time,gamma,lambda) tuples that define the annealing schedule (optional=[(30,0.0,1.0)])

runonce

Type: Microsoft.FSharp.CoreFSharpOptionBoolean

true=run the simulation once and perform repeated 'virtual measurements'; false=run the simulation and measure each time (optional=false)

decohereModel

Type: Microsoft.FSharp.CoreFSharpOptionFSharpListTupleFS Unit, Double

Decoherence probability per qubit per timestep [(gate,prob) list] optional=None

⊿ See Also

Reference

Spin Class Microsoft.Research.Liquid Namespace

SpinPeek Method

Peeks at the current Hamiltonian matrix. This is based on the most recently grown circuit.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# _____Copy _____
member Peek :
    anneal : FSharpOption<float[]> -> CSMat
```

Parameters

anneal

Type: Microsoft.FSharp.CoreFSharpOptionDouble

An optional array of annealing values to use. The default is to use the most recent annealing values.

Return Value

Type: CSMat

A sparse matrix representation of the Hamiltonian

⊿ See Also

Reference Spin Class Microsoft.Research.Liquid Namespace

SpinPrep Method

Prepares the qubit state for a run. All qubits are resets and initialized to the ground state.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)



Microsoft.Research.Liquid Namespace

SpinRun Method

Runs the simulation. This is a highly optimized implementation.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F#
Copy
member Run :
    dt : int *
    schedule : FSharpList<Tuple<int, float[]>
    gp : FSharpOption<GrowPars> -> int
```

Parameters

dt

Type: SystemInt32

The number of timesteps to run.

schedule

Type: Microsoft.FSharp.CollectionsFSharpListTupleInt32, Double

The annealing schedule to use. Each entry in the list is a tuple whose first entry is a time step and whose second entry is an array of annealing values. Annealing values for time steps in between entries are computed by linearly interpolating between those for the previous and next entries. The list must be in ascneding order by time step.

gp

Type: **Microsoft.FSharp.CoreFSharpOptionGrowPars** Optional grow parameters for the resulting circuit. The default is GrowGates with maxWires of 11. See GrowPars for more details.

Return Value

Type: Int32

The number of decoherence events which occured during the step. Note that this will be zero unless a decoherence model has been set on this instance.

⊿ See Also

Reference Spin Class Microsoft.Research.Liquid Namespace

SpinStep Method

Performs a single timestep of simulation.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

angles

Type: SystemDouble

The vector of annealing coefficients to be used in this step.

gр

```
Type: Microsoft.FSharp.CoreFSharpOptionGrowPars
Optional grow parameters for the resulting circuit. The default is
GrowGates with maxWires of 11. See GrowPars for more
details.
```

Return Value

Type: Int32

The number of decoherence events which occured during the step. Note that this will be zero unless a decoherence model has been set on this instance.

⊿ See Also

Reference

Spin Class Microsoft.Research.Liquid Namespace

SpinTest Method

Overload List

	Name	Descrip
≕Ŷ S	Test(String, Int32, Int32, FSharpListTupleInt32, Double, Int32, Spin, FSharpOptionBoolean, FSharpOptionFSharpListTupleFSharpFuncQubit, Unit, Double)	Execute set of simulatic runs for spin Hamiltor
= ≬ S	Test(String, Int32, Int32, DictionaryInt32, Double, DictionaryTupleInt32, Int32, Double, FSharpListTupleInt32, Double, Double, Int32, FSharpOptionBoolean, FSharpOptionFSharpListTupleFSharpFuncQubit, Unit, Double)	Execute set of simulatic runs for spin Hamiltor

Тор

⊿ See Also

Reference Spin Class Microsoft.Research.Liquid Namespace

SpinTest Method (String, Int32, Int32 FSharpListTupleInt32, Double, Int32 FSharpOptionBoolean, FSharpOptionFSharpListTupleFSha Unit, Double)

Executes a set of simulation runs for a spin Hamiltonian.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# Copy
static member Test :
    tag : string *
    repeats : int *
    trotter : int *
    schedule : FSharpList<Tuple<int, float[]>
    res : int *
    spin : Spin *
    runonce : FSharpOption<bool> *
    decohereModel : FSharpOption<FSharpList<1</pre>
```

Parameters

tag

Type: SystemString The output label for logging. *repeats*

Type: SystemInt32

The number of simulations to run.

trotter

Type: SystemInt32

The Trotter number to use.

schedule

Type: Microsoft.FSharp.CollectionsFSharpListTupleInt32, Double

The annealing schedule to use. Each entry in the list is a tuple whose first entry is a time step and whose second entry is an array of annealing values. Annealing values for time steps in between entries are computed by linearly interpolating between those for the previous and next entries. The list must be in ascneding order by time step.

res

Type: SystemInt32

The resolution of the simulation, in time steps. Larger values may increase speed but will reduce the granularity of output. Note that this must evenly divide the final time in the annealing schedule.

spin

Type: Microsoft.Research.LiquidSpin

The actual Hamiltonian to be simulated.

runonce

Type: Microsoft.FSharp.CoreFSharpOptionBoolean

An option to only run the simulation once and perform repeated 'virtual measurements'. The default is false, which means to run the simulation and measure each time.

decohereModel

Type: Microsoft.FSharp.CoreFSharpOptionFSharpListTupleFS Unit, Double

An optional decoherence model to use for this simulation. See **Hamiltonian.decohereModel** for more information. The default is no decoherence.

⊿ See Also

Reference Spin Class Test Overload Microsoft.Research.Liquid Namespace

SpinTest Method (String, Int32, Int32 DictionaryInt32, Double, DictionaryT Int32, Double, FSharpListTupleInt32 Double, Int32, FSharpOptionBooleau FSharpOptionFSharpListTupleFSha Unit, Double)

Executes a set of simulation runs for a spin Hamiltonian.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

tag
Type: SystemString

The output label for logging.

repeats

Type: SystemInt32

The number of simulations to run.

trotter

Type: SystemInt32

The Trotter number to use.

hs

Type: System.Collections.GenericDictionaryInt32, Double A Dictionary that maps a qubit id to that qubit's bias strength (Z term coefficient).

Js

Type: System.Collections.GenericDictionaryTupleInt32, Int32, Double

A Dictionary that maps a pair of qubit id's to the pair's coupling strength (ZZ coefficient).

gammalambda

Type: Microsoft.FSharp.CollectionsFSharpListTupleInt32, Double, Double

List of (time,gamma,lambda) tuples that define the annealing schedule. Each entry in the list is a tuple whose first entry is a time step and whose second entry is an array of annealing values. Annealing values for time steps in between entries are computed by linearly interpolating between those for the previous and next entries. The list must be in ascneding order by time step.

res

Type: SystemInt32

The resolution of the simulation, in time steps. Larger values may increase speed but will reduce the granularity of output. Note that this must evenly divide the final time in the annealing schedule.

runonce

Type: Microsoft.FSharp.CoreFSharpOptionBoolean

An option to only run the simulation once and perform repeated 'virtual measurements'. The default is false, which means to run

the simulation and measure each time.

decohereModel

Type: Microsoft.FSharp.CoreFSharpOptionFSharpListTupleFS Unit, Double

An optional decoherence model to use for this simulation. See **Hamiltonian.decohereModel** for more information. The default is no decoherence.

⊿ See Also

Reference Spin Class Test Overload Microsoft.Research.Liquid Namespace

SpinTerm Class

A single term in a Spin Hamiltonian.

Inheritance Hierarchy

SystemObject Microsoft.Research.LiquidSpinTerm

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

```
[<SerializableAttribute>]
type SpinTerm = class end
```

The SpinTerm type exposes the following members.

Constructors

	Name	Description
≕ \$	SpinTerm(Int32, FSharpFuncDouble, FSharpFuncFSharpListQubit, Unit, Double)	Initializes a new SpinTerm instance.
≕ \$	SpinTerm(Int32, FSharpFuncDouble, FSharpFuncFSharpListQubit, Unit, FSharpListInt32, Double)	Initializes a new SpinTerm instance.

Тор

Properties

	Name	Description
*	Amplitude	The numerical coefficient ('strength') leading this term.
*	Schedule	The numerical id (zero-based) of the annealing schedule corresponding to this term.

Тор

▲ Methods

	Name	Description
=♥	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
=∳	GetHashCode	Serves as the default hash function. (Inherited from Object.)
=0	GetType	Gets the Type of the current instance. (Inherited from Object.)
=∳	ToString	Returns a string that represents the current object. (Inherited from Object.)

Тор

⊿ See Also

Reference

Microsoft.Research.Liquid Namespace

SpinTerm Constructor

Overload List

	Name	Description
≡♥	SpinTerm(Int32, FSharpFuncDouble, FSharpFuncFSharpListQubit, Unit, Double)	Initializes a new SpinTerm instance.
≓ ∲	SpinTerm(Int32, FSharpFuncDouble, FSharpFuncFSharpListQubit, Unit, FSharpListInt32, Double)	Initializes a new SpinTerm instance.

Тор

⊿ See Also

Reference SpinTerm Class Microsoft.Research.Liquid Namespace

SpinTerm Constructor (Int32, FSharpFuncDouble, FSharpFuncFSharpListQubit, Unit, Double)

Initializes a new SpinTerm instance.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#	Сору
new :	
	s : int *
	<pre>o : FSharpFunc<float, fsharpfunc<fsharpli<="" pre=""></float,></pre>
	a : float -> SpinTerm
[4]	

Parameters

S

Type: SystemInt32

The numerical id, zero-based, of the annealing schedule corresponding to this term.

0

```
Type: Microsoft.FSharp.CoreFSharpFuncDouble,
FSharpFuncFSharpListQubit, Unit
```

A function which performs the operation of this term over a given angle.

а

Type: SystemDouble The numerical coefficient ('strength') leading this term.

⊿ See Also

Reference SpinTerm Class SpinTerm Overload Microsoft.Research.Liquid Namespace

SpinTerm Constructor (Int32, FSharpFuncDouble, FSharpFuncFSharpListQubit, Unit, FSharpListInt32, Double)

Initializes a new SpinTerm instance.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

S

Type: SystemInt32

The numerical id, zero-based, of the annealing schedule corresponding to this term.

0

Type: Microsoft.FSharp.CoreFSharpFuncDouble, FSharpFuncFSharpListQubit, Unit

A function which performs the operation of this term over a given angle.

idx

Type: Microsoft.FSharp.CollectionsFSharpListInt32

A list of integer indices specifying the qubits to apply this term to.

а

Type: SystemDouble

The numerical coefficient ('strength') leading this term.

⊿ See Also

Reference SpinTerm Class SpinTerm Overload Microsoft.Research.Liquid Namespace

SpinTerm Properties

The SpinTerm type exposes the following members.

▲ Properties

	Name	Description
*	Amplitude	The numerical coefficient ('strength') leading this term.
*	Schedule	The numerical id (zero-based) of the annealing schedule corresponding to this term.

Тор

⊿ See Also

Reference SpinTerm Class Microsoft.Research.Liquid Namespace

SpinTermAmplitude Property

The numerical coefficient ('strength') leading this term.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____Copy _____ member Amplitude : float with get _____ Property Value Type: Double See Also

Reference SpinTerm Class Microsoft.Research.Liquid Namespace

SpinTermSchedule Property

The numerical id (zero-based) of the annealing schedule corresponding to this term.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

___ Сору __

member Schedule : int with get

Property Value Type: Int32

⊿ See Also

Reference SpinTerm Class Microsoft.Research.Liquid Namespace

SpinTerm Methods

The SpinTerm type exposes the following members.

Methods

	Name	Description
=♥	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
=0	GetHashCode	Serves as the default hash function. (Inherited from Object.)
=∳	GetType	Gets the Type of the current instance. (Inherited from Object.)
=∳	ToString	Returns a string that represents the current object. (Inherited from Object.)

Тор

⊿ See Also

Reference SpinTerm Class Microsoft.Research.Liquid Namespace

Stabilizer Class

A stabilizer-based simulator based on CHP by Scott Aaronson and Daniel Gottesman. See arXiv:guant-ph/0406196 for more details.

▲ Inheritance Hierarchy

SystemObject Microsoft.Research.LiquidStabilizer

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

◢

F# Copy [<SerializableAttribute>]

```
type Stabilizer = class end
```

The Stabilizer type exposes the following members.

Constructors

	Name	Description
=0	Stabilizer	Creates a tableau to run a circuit in the stabilizer simulator.
Тор		
Propert	ies	
	Name	Description
	Item	The current measured value for a gubit.

Тор

Methods

	Name	Description
=♥	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
=0	Gaussian	Performs a Gaussian elimination to put the tableau in quasi upper triangluar form.
=♥	GetHashCode	Serves as the default hash function. (Inherited from Object.)
≡ ⊘	GetType	Gets the Type of the current instance. (Inherited from Object.)
≡Q	Run	Runs the circuit.
≡∳	ShowState	Dumps the stabilizer tableau.
≡ Q	ToString	Returns a string that represents the current object. (Inherited from Object.)

Тор

⊿ See Also

Reference Microsoft.Research.Liquid Namespace

Stabilizer Constructor

Creates a tableau to run a circuit in the stabilizer simulator.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

circ

Type: Microsoft.Research.LiquidCircuit The crcuit that will be run in stabilizer mode.

ket

Type: Microsoft.Research.LiquidKet The state vector that will be used for execution.

▲ See Also

Reference Stabilizer Class Microsoft.Research.Liquid Namespace

Stabilizer Properties

The Stabilizer type exposes the following members.

▲ Properties

		Name	Description
	₩	Item	The current measured value for a qubit.
	Тор		
4	₄ See Also		
Reference Stabilizer Class Microsoft.Research.Liquid Namespace			

StabilizerItem Property

The current measured value for a qubit.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____Copy ____Copy _____Copy ____COPY _____COPY _____COPY _____COPY ____COPY _____COPY _____COPY _____COPY _____COPY ____COPY ___COPY __COPY __COPY ___COPY __COPY _COPY __COPY __COPY _COPY __COPY __COPY _COPY _COPY _COP

Parameters

i

Type: SystemInt32 The index of the qubit in the simulator's state vector.

Property Value Type: TupleBoolean, Bit

Remarks

This property has a tuple value. The first item is a flag indicating whether the last measurement was random or forced, and the second item is the result of the last measurement as a Bit. As with any qubit, the Bit will be Unknown if the qubit has interacted since it was last measured.

⊿ See Also

Reference Stabilizer Class Microsoft.Research.Liquid Namespace

Stabilizer Methods

The Stabilizer type exposes the following members.

▲ Methods

	Name	Description
÷	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
. ≕ ∲	Gaussian	Performs a Gaussian elimination to put the tableau in quasi upper triangluar form.
≓ ©	GetHashCode	Serves as the default hash function. (Inherited from Object.)
≓ ©	GetType	Gets the Type of the current instance. (Inherited from Object.)
≡ ∲	Run	Runs the circuit.
≡∳	ShowState	Dumps the stabilizer tableau.
÷	ToString	Returns a string that represents the current object. (Inherited from Object.)

Тор

⊿ See Also

Reference Stabilizer Class Microsoft.Research.Liquid Namespace

StabilizerGaussian Method

Performs a Gaussian elimination to put the tableau in quasi upper triangluar form.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Syntax F# ______Copy ____ member Gaussian : unit -> unit See Also Reference Stabilizer Class Microsoft.Research.Liquid Namespace

StabilizerRun Method

Runs the circuit.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

gaussian : FSharpOption<bool> -> unit

Parameters

circuit

Type: Microsoft.FSharp.CoreFSharpOptionCircuit

An optional different circuit to run. This allows the tableau to be reused for different circuits. The default is to used the circuit the tableau was created with.

dumpAll

Type: Microsoft.FSharp.CoreFSharpOptionBoolean

An option indicating that the tableau state should be written to the log after each gate application. The default is false, indicating not to log.

gaussian

Type: Microsoft.FSharp.CoreFSharpOptionBoolean

An option indicating that the tableau state should be simplified using Gaussian elimination before being logged. This is ignored if *dumpall* is false. The default is false, indicating not to perform Gaussian elimination.

⊿ See Also

Reference Stabilizer Class Microsoft.Research.Liquid Namespace

StabilizerShowState Method

Dumps the stabilizer tableau.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# Copy
member ShowState :
    f : FSharpFunc<int, FSharpFunc<string, Ur
    level : int -> unit
```

Parameters

f

Type: Microsoft.FSharp.CoreFSharpFuncInt32, FSharpFuncString, Unit

The output function to use. A common output function is showLogInd.

level

Type: SystemInt32 The indentation level.

▲ See Also

Reference Stabilizer Class Microsoft.Research.Liquid Namespace

Steane7 Class

Implementation of a Steane 7-bit quantum error correcting code, [[7,1,3]], based on the QECC class.

▲ Inheritance Hierarchy

```
SystemObject Microsoft.Research.LiquidQECC
Microsoft.Research.LiquidSteane7
```

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



The Steane7 type exposes the following members.

Constructors

	Name	Description
= Q	Steane7	Constructs a Steane code implementation for a circuit.

Тор

Properties

	Name	Description
*	Circuit	The error-correcting circuit built by Compile. The circuit will be built now if it hasn't already been. (Inherited from QECC.)
	Ket	The state vector for the compiled code. (Inherited from QECC.)
*	NumFixed	The number of syndrome fixups performed.

Тор

▲ Methods

	Name	Description
- = Q	Compile	Compiles the target circuit into an error-correcting version. (Inherited from QECC.)
- = \$	Decode	Decodes a set of measured physical qubits to get the measured value for a logical qubit. (Overrides QECCDecode(FSharpListQubit).)
≓ ∳	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
≓Q	GetHashCode	Serves as the default hash function. (Inherited from Object.)
=♦	GetMeasured	Gets the measured values of the

		physical qubits that make up a logical qubit and returns them combined into a single integer, one bit per qubit. (Inherited from QECC.)
= 0	GetType	Gets the Type of the current instance. (Inherited from Object.)
= Q	Inject	Injects dephasing errors with the given probability into the error- correcting circuit. (Inherited from QECC.)
= \	Log2Phys	Gets the physical qubits that make up a logical qubit. (Inherited from QECC.)
= \$	Prep	A gate function that prepares a logical 0> qubit. This gets compiled into the error-correcting circuit by the Compile method. (Overrides QECCPrep(FSharpListQubit).)
= 0	Replace	Gets a replacement physical gate for an input logical gate. The replacement may wrap a full Circuit. (Inherited from QECC.)
≓Ŵ	Syndrome	A gate function that measures the physical qubits for a single logical qubit and applies any necessary corrections. This gets compiled into the error-correcting circuit by the Compile method. (Overrides

		QECCSyndrome(FSharpListQubit).)
≓ ≬ S	Test1	Tests the Steane7 QECC on a single logical qubit with forced X, Y, and Z error injections.
≡ ©	ToString	Returns a string that represents the current object. (Inherited from Object.)

Тор

▲ Remarks

This code uses 7 physical qubits per logical qubit. It also requires 6 ancillae.

⊿ See Also

Reference Microsoft.Research.Liquid Namespace Microsoft.Research.LiquidQECC

Steane7 Constructor

Constructs a Steane code implementation for a circuit.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Steane7 Properties

The Steane7 type exposes the following members.

Properties

		Name	Description
		Circuit	The error-correcting circuit built by Compile. The circuit will be built now if it hasn't already been. (Inherited from QECC.)
		Ket	The state vector for the compiled code. (Inherited from QECC.)
	*	NumFixed	The number of syndrome fixups performed.
	Тор		
4	See Als	0	
	Reference Steane7 Clas Microsoft.Res	s search.Liquid	d Namespace

Steane7NumFixed Property

The number of syndrome fixups performed.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# ______ Copy __ member NumFixed : int with get, set Property Value Type: Int32 • See Also Reference Steane7 Class

Microsoft.Research.Liquid Namespace

Steane7 Methods

The Steane7 type exposes the following members.

Methods

	Name	Description
≓Ŵ	Compile	Compiles the target circuit into an error-correcting version. (Inherited from QECC.)
≓ ©	Decode	Decodes a set of measured physical qubits to get the measured value for a logical qubit. (Overrides QECCDecode(FSharpListQubit).)
≡♥	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
≡ ©	GetHashCode	Serves as the default hash function. (Inherited from Object.)
≓ \$	GetMeasured	Gets the measured values of the physical qubits that make up a logical qubit and returns them combined into a single integer, one bit per qubit. (Inherited from QECC.)
≡∳	GetType	Gets the Type of the current instance. (Inherited from Object.)
= Q	Inject	Injects dephasing errors with the given probability into the error- correcting circuit. (Inherited from QECC.)
--------------	----------	---
≓∳	Log2Phys	Gets the physical qubits that make up a logical qubit. (Inherited from QECC.)
=♥	Prep	A gate function that prepares a logical 0> qubit. This gets compiled into the error-correcting circuit by the Compile method. (Overrides QECCPrep(FSharpListQubit).)
-=•	Replace	Gets a replacement physical gate for an input logical gate. The replacement may wrap a full Circuit. (Inherited from QECC.)
=	Syndrome	A gate function that measures the physical qubits for a single logical qubit and applies any necessary corrections. This gets compiled into the error-correcting circuit by the Compile method. (Overrides QECCSyndrome(FSharpListQubit).)
= 0 S	Test1	Tests the Steane7 QECC on a single logical qubit with forced X, Y, and Z error injections.
≡♥	ToString	Returns a string that represents the current object. (Inherited from Object.)

Тор

⊿ See Also

Reference Steane7 Class Microsoft.Research.Liquid Namespace

Steane7Decode Method

Decodes a set of measured physical qubits to get the measured value for a logical qubit.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

qs

Type: **Microsoft.FSharp.CollectionsFSharpList**Qubit The physical qubits to decode. They must already have been measured.

Return Value

Type: TupleBit, Int32

A tuple containing the logical measured value of the logical qubit, either Zero or One, and the Hamming distance from the physical state to the code space.

⊿ See Also

Reference Steane7 Class Microsoft.Research.Liquid Namespace

Steane7Prep Method

A gate function that prepares a logical |0> qubit. This gets compiled into the error-correcting circuit by the Compile method.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Parameters

qs

Type: Microsoft.FSharp.CollectionsFSharpListQubit

The physical qubits for the logical qubit. This contains both data and syndrome qubits, but no ancillae.

Copy

⊿ See Also

Reference Steane7 Class Microsoft.Research.Liquid Namespace

Steane7Syndrome Method

A gate function that measures the physical qubits for a single logical qubit and applies any necessary corrections. This gets compiled into the error-correcting circuit by the Compile method.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

qs

Type: **Microsoft.FSharp.CollectionsFSharpListQubit** The ancilla qubits followed by physical qubits for the logical qubit. The number of ancillae was passed to the QECC constructor.

⊿ See Also

Reference Steane7 Class Microsoft.Research.Liquid Namespace

Steane7Test1 Method

Tests the Steane7 QECC on a single logical qubit with forced X, Y, and Z error injections.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____Copy ____
static member Test1 : unit -> unit

Remarks

The test verifies that the Steane 7 code properly decodes a prepared logical qubit in the face of all possible single-qubit dephasing errors on each of the 7 physical qubits.

⊿ See Also

Reference Steane7 Class Microsoft.Research.Liquid Namespace

Tests Class

A collection of sample Liquid simulations and tests, plus some utility routines to make it easier to write new samples.

▲ Inheritance Hierarchy

```
SystemObject Microsoft.Research.LiquidTests
```

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



The Tests type exposes the following members.

Methods

	Name	Description
= 0 S	Big	Runs an entanglement test for state vectors from 16 to 22 qubits in size.
= 0 S	Chem	Runs a simple quantum chemistry simulation. This is the same as the ChemFull sample, with common values for the detailed parameters: test 0, 32 Trotter steps, first-order

		Trotter, 28 bits of accuracy, and no additional options. See the Users Manual for more information.
= ≬ S	ChemFull	Runs a quantum chemistry simulation. See the Users Manual for more information.
= 0 S	Correct	Test various permutations for correctness using teleport
= 0 S	EIGS	Validates that LAPACK is properly installed.
= ≬ S	Entangle1	Runs a simple gate sequence that entangles all the qubits in the state vector.
≓∳ S	Entangle2	Runs a simple gate sequence that entangles all the qubits in the state vector. The sequence is run three different ways to demonstrate the difference in timings.
= ≬ S	Entangles	Runs 100 entanglement tests on 16 qubits to show the statistics on the bits measured.
= ≬ S	EntEnt	Demonstrates a couple of simple entanglement entropy calculations.
= \$ \$	EPR	Renders a small EPR circuit that entangles two qubits.
= ≬ S	Ferro	Simulates a ferromagnetic chain using a first-quantized Hamiltonian.

⊴ ∳ S	JointCNOT	Demonstrates and tests various ways of implementing CNOT from joint measurements and single- qubit Clifford gates.
≓∳ S	Kraus	Show how to add noise to Teleport with Kraus operators
:∳ S	Noise1	Demonstrates the use of a complex noise model with error correction. See the Users Manual for more information.
= ≬ S	NoiseAmp	Demonstrates a complex noise model. See the Users Manual for more information.
= ≬ S	QECC	Demonstrates some examples of error correction and stabilizer simulation. See the Users Manual for details.
=ŷ S	QFTbench	Benchmarks the Quantum Fourier Transform at the heart of the Shor algorithm.
= 0 S	QLSA	Demonstrates the Quantum Linear Algebra algorithm from Harrow, Hassidim, and Lloyd. See the Users Manual for more information.
≕ \$ S	QuAM	Demonstrates the Quantum Associative Memory algorithm from Ventura and Martinez. See the Users Manual for more information.

⊴\$	QWalk	Demonstrates the Quantum PageRank algorithm from Paparo and Martin-Delgado. See the Users Manual for more information.
≡ŷ S	Ramsey33	Solves for the (3,3) Ramsey number. See this paper for details.
≡ŷ S	SG	Simulates a spin glass using a first-quantized Hamiltonian.
≡ŷ S	Shor	Runs the classic Shor factoring algorithm.
≓∳ S	Steane7	Validates that the Steane 7 code is correct. See Steane7.Test1 for more details.
≡ŷ S	Teleport	Renders and runs the classic quantum teleportation algorithm.
≡ŷ S	TSP	Solves the traveling salesman problem.
≓∳ S	RenderTest	Renders and dumps test circuits in a variety of formats and detail levels.

Тор

⊿ See Also

Reference Microsoft.Research.Liquid Namespace

Tests Methods

The Tests type exposes the following members.

Methods

	Name	Description
= 0 S	Big	Runs an entanglement test for state vectors from 16 to 22 qubits in size.
≕∲ S	Chem	Runs a simple quantum chemistry simulation. This is the same as the ChemFull sample, with common values for the detailed parameters: test 0, 32 Trotter steps, first-order Trotter, 28 bits of accuracy, and no additional options. See the Users Manual for more information.
= 0 S	ChemFull	Runs a quantum chemistry simulation. See the Users Manual for more information.
= 0 S	Correct	Test various permutations for correctness using teleport
≓Ŷ S	EIGS	Validates that LAPACK is properly installed.
= 0 S	Entangle1	Runs a simple gate sequence that entangles all the qubits in the state vector.

⊴ ∳ S	Entangle2	Runs a simple gate sequence that entangles all the qubits in the state vector. The sequence is run three different ways to demonstrate the difference in timings.
= 0 S	Entangles	Runs 100 entanglement tests on 16 qubits to show the statistics on the bits measured.
= ≬ S	EntEnt	Demonstrates a couple of simple entanglement entropy calculations.
= 0 S	EPR	Renders a small EPR circuit that entangles two qubits.
= ≬ S	Ferro	Simulates a ferromagnetic chain using a first-quantized Hamiltonian.
= 0 S	JointCNOT	Demonstrates and tests various ways of implementing CNOT from joint measurements and single- qubit Clifford gates.
= ≬ S	Kraus	Show how to add noise to Teleport with Kraus operators
= 0 S	Noise1	Demonstrates the use of a complex noise model with error correction. See the Users Manual for more information.
: ∳ S	NoiseAmp	Demonstrates a complex noise model. See the Users Manual for more information.

≡≬ S	QECC	Demonstrates some examples of error correction and stabilizer simulation. See the Users Manual for details.
≓ ≬ S	QFTbench	Benchmarks the Quantum Fourier Transform at the heart of the Shor algorithm.
= ≬ S	QLSA	Demonstrates the Quantum Linear Algebra algorithm from Harrow, Hassidim, and Lloyd. See the Users Manual for more information.
= ∲ S	QuAM	Demonstrates the Quantum Associative Memory algorithm from Ventura and Martinez. See the Users Manual for more information.
= ≬ S	QWalk	Demonstrates the Quantum PageRank algorithm from Paparo and Martin-Delgado. See the Users Manual for more information.
= 0 S	Ramsey33	Solves for the (3,3) Ramsey number. See this paper for details.
= \$	SG	Simulates a spin glass using a first-quantized Hamiltonian.
= 0 S	Shor	Runs the classic Shor factoring algorithm.
= ≬ S	Steane7	Validates that the Steane 7 code is correct. See Steane7.Test1 for

		more details.
= 0 S	Teleport	Renders and runs the classic quantum teleportation algorithm.
= 0 S	TSP	Solves the traveling salesman problem.
⊧ ≬ S	RenderTest	Renders and dumps test circuits in a variety of formats and detail levels.

Тор

⊿ See Also

Reference Tests Class Microsoft.Research.Liquid Namespace

Tests___Big Method

Runs an entanglement test for state vectors from 16 to 22 qubits in size.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)



Tests_Chem Method

Runs a simple quantum chemistry simulation. This is the same as the ___ChemFull sample, with common values for the detailed parameters: test 0, 32 Trotter steps, first-order Trotter, 28 bits of accuracy, and no additional options. See the Users Manual for more information.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

static member __Chem :
 mol : string -> unit

Parameters

mol

Type: SystemString

The name of the molecule to simulate. Entering an empty string, "", will display the list of available molecules.

Copy

⊿ See Also

Reference

Tests Class

Microsoft.Research.Liquid Namespace

Tests ChemFull Method

Runs a quantum chemistry simulation. See the Users Manual for more information.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

```
static member __ChemFull :
    mol : string *
    test : int *
    opts : string *
    trot : int *
    bits : int *
    order : int -> unit
```

Parameters

mol

Type: SystemString

The name of the molecule to simulate. Entering an empty string, "", will display the list of available molecules.

Copy

test

Type: SystemInt32

The test number to run, from the .dat file.

opts

Type: SystemString

A string of any options you want to set. See the Users Manual for more information.

trot

Type: SystemInt32

The Trotter parameter; that is, number of Trotter steps per time step.

bits

Type: SystemInt32

The number of bits of accuracy desired in the phase estimation. *order*

Type: SystemInt32

The order of Trotter approximation to use, 1 or 2.

⊿ See Also

Reference

Tests Class Microsoft.Research.Liquid Namespace

Tests_Correct Method

Test various permutations for correctness using teleport

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Tests__EIGS Method

Validates that LAPACK is properly installed.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Tests__Entangle1 Method

Runs a simple gate sequence that entangles all the qubits in the state vector.

Copy

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

static member __Entangle1 :
 entSiz : int -> unit

Parameters

entSiz

Type: SystemInt32

The number of qubits desired in the state vector.

▲ Remarks

This sample displays detailed timing and memory usage information, and so allows you to see how Liquid scales as the state vector size grows.

⊿ See Also

Reference Tests Class Microsoft.Research.Liquid Namespace

Tests__Entangle2 Method

Runs a simple gate sequence that entangles all the qubits in the state vector. The sequence is run three different ways to demonstrate the difference in timings.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

entSiz

Type: SystemInt32

The number of qubits desired in the state vector.

Remarks

This test executes the same gate sequence in three different ways:

- As a simple sequence of gates. This is essentially the same as the <u>Entangle1</u> sample.
- As a compiled circuit.
- As a compiled circuit that has been optimized by calling GrowGates.

⊿ See Also

Reference Tests Class Microsoft.Research.Liquid Namespace

Tests__Entangles Method

Runs 100 entanglement tests on 16 qubits to show the statistics on the bits measured.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)



Tests__EntEnt Method

Demonstrates a couple of simple entanglement entropy calculations.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Tests__EPR Method

Renders a small EPR circuit that entangles two qubits.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Tests__Ferro Method

Simulates a ferromagnetic chain using a first-quantized Hamiltonian.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# static member ___Ferro : full : bool * runonce : bool -> unit

Parameters

full

Type: SystemBoolean

Whether to run all chain variations (isolated, ferromagnetic, antiferromagnetic, freeze up, freese down, and freeze up/down) or just the last variation, freeze up/down.

Copy

runonce

Type: SystemBoolean

For each variation, whether to run the circuit once and then do "virtual sampling" from the state vector, or to run the circuit and perform full simulated measurements each time.

⊿ See Also

Reference

Tests Class Microsoft.Research.Liquid Namespace

Tests__JointCNOT Method

Demonstrates and tests various ways of implementing CNOT from joint measurements and single-qubit Clifford gates.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)



Tests_Kraus Method

Show how to add noise to Teleport with Kraus operators

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#	_ Сору
<pre>static memberKraus : nRuns : int * probAD : float * probDP : float * verbose : bool -> unit</pre>	
Parameters	
nRuns Type: SystemInt32 How many runs to gather statistics over probAD Type: SystemDouble Probability of Amplitude Damping on any single qubit probDP Type: SystemDouble Probability of Depolarizing noise on any single qubit verbose Type: SystemBoolean Output detailed stats and drawings	
See Also	

Reference

Tests Class Microsoft.Research.Liquid Namespace

Tests__Noise1 Method

Demonstrates the use of a complex noise model with error correction. See the Users Manual for more information.

Copy

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#
static member __Noise1 :
 depth : int *
 iters : int *
 prob : float -> unit

Parameters

depth

Type: SystemInt32

The number of idle gates to include in the circuit.

iters

Type: SystemInt32

The number of executions to run, for statistical purposes.

prob

Type: SystemDouble

The probability of an error occurring.

⊿ See Also

Reference Tests Class Microsoft.Research.Liquid Namespace

Tests__NoiseAmp Method

Demonstrates a complex noise model. See the Users Manual for more information.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)


Tests__QECC Method

Demonstrates some examples of error correction and stabilizer simulation. See the Users Manual for details.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)



Tests__QFTbench Method

Benchmarks the Quantum Fourier Transform at the heart of the Shor algorithm.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)



Tests__QLSA Method

Demonstrates the Quantum Linear Algebra algorithm from Harrow, Hassidim, and Lloyd. See the Users Manual for more information.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)



Tests__QuAM Method

Demonstrates the Quantum Associative Memory algorithm from Ventura and Martinez. See the Users Manual for more information.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)



Tests__QWalk Method

Demonstrates the Quantum PageRank algorithm from Paparo and Martin-Delgado. See the Users Manual for more information.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

which

Type: SystemString

The web graph to use. There are three built-in graphs: tiny, tree, and graph. Alternatively, the path to a .graph file may be provided.

⊿ See Also

Reference Tests Class Microsoft.Research.Liquid Namespace

Tests__Ramsey33 Method

Solves for the (3,3) Ramsey number. See this paper for details.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Tests__SG Method

Simulates a spin glass using a first-quantized Hamiltonian.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Tests__Shor Method

Runs the classic Shor factoring algorithm.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____Copy ____Copy _____Copy ____COPY _____COPY _____COPY ____COPY _____COPY _____COPY _____COPY _____COPY ____COPY ___COPY ____COPY ____COPY ____COPY ____COPY ____COPY ____COPY ____COPY ___COPY __COPY __COPY ___COPY __COPY __COPY __COPY __COPY __COPY __COPY __COPY __COPY __CO

Parameters

Ν

Type: SystemInt32 The number to factor. doCirc

Type: SystemBoolean Whether or not to optimize the circuit.

⊿ See Also

Reference Tests Class Microsoft.Research.Liquid Namespace

Tests__Steane7 Method

Validates that the Steane 7 code is correct. See <u>Steane7.Test1</u> for more details.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)



Tests___Teleport Method

Renders and runs the classic quantum teleportation algorithm.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Tests__TSP Method

Solves the traveling salesman problem.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

nCities

Type: SystemInt32 The number of cities to include in the map, from 5 to 8.

⊿ See Also

Reference Tests Class Microsoft.Research.Liquid Namespace

TestsRenderTest Method

Renders and dumps test circuits in a variety of formats and detail levels.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

static member RenderTest :
 name : string *
 circ : Circuit *
 ket : Ket -> unit

Parameters

name

Type: SystemString The base name for the output files.

circ

Type: Microsoft.Research.LiquidCircuit The Circuit to render.

ket

Type: Microsoft.Research.LiquidKet The state vector for the circuit.

Remarks

Three versions of the circuit will be dumped to the log file:

- The base circuit.
- The circuit aggressively folded to maximize parallelism.
- The circuit with gates grown to a maximum of 11 inputs, and

then folded.

Four pairs of graphics files will be created. For each version, both an SVG version, with a .htm extension, and a Tik-Z version, with a .tex, will be created.

- The base circuit.
- The circuit normally folded to use "easy" parallelism.
- The circuit aggressively folded to maximize parallelism.
- The circuit with gates grown to a maximum of 11 inputs, and then folded.

⊿ See Also

Reference Tests Class Microsoft.Research.Liquid Namespace

Util Class

General utilities used by the rest of the system

▲ Inheritance Hierarchy

SystemObject Microsoft.Research.LiquidUtil

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

```
[<AbstractClassAttribute>]
[<SealedAttribute>]
type Util = class end
```

The Util type exposes the following members.

▲ Properties

	Name	Description
🖹 S	copyright	Copyright notice for Liquid.
🖀 2	interactive	
🖹 S	outputPrefix	Global override for the prefix used for output functions (show and variants).
📽 S	sqrt2	Highly accurate square root of 2, for use building unitary matrices.

Тор

▲ Methods

	Name	Description
≓ ù S	show	Shows a line of output. This is a command line-callable routine.
≓Ŷ S	getFlgPar	Extracts a boolean macro definition from a command option dictionary.
≓∳ S	getFltPar	Extracts a float macro definition from a command option dictionary.
=\$ S	getIntPar	Extracts an integer macro definition from a command option dictionary.
=\$ S	getIntsPar	Extracts an integer list macro definition from a command option dictionary.
=\$ S	getStrPar	Extracts a string macro definition from a command option dictionary.
=\$ S	Normal	Add a Normal distribution to the System.Random class. This is an extension method and may be used as if it were an instance method on Random.
≓ ≬ S	PermAry	Create a random permutation array (call .permute with

		results) This is an extension method and may be used as if it were an instance method on Random.
≓ ≬ S	procStats	Gets current process memory usage statistics.
≓ ≬ S	showa	Shows a line of output to the console and the log file.
= ∲	showBareInd	Shows an indented line of output, with no prefix, to the console and log file. This routine is used by or with various Dump() routines.
=\$ S	showBareLogInd	Shows an indented line of output, with no prefix, to the log file. This routine is used by or with various Dump() routines.
= 0 S	showDump	Dumps out a string with with an optional indentation.
≓≬ S	showInd	Shows an indented line of output to the console and log file. This routine is used by or with various Dump() routines.
≓ ≬ S	showLoga	Shows a line of output to the log file.
=\$ S	showLogInd	Shows an indented line of output to the log file. This routine is used by or with various Dump() routines.
= ≬ S		

showProcStats	Print process memory usage statistics to the console and log file.

Тор

⊿ See Also

Reference

Microsoft.Research.Liquid Namespace

Util Properties

The Util type exposes the following members.

▲ Properties

	Name	Description	
🖹 S	copyright	Copyright notice for Liquid.	
🖹 S	interactive		
🚰 S	outputPrefix	Global override for the prefix used for output functions (show and variants).	
🕈 s	sqrt2	Highly accurate square root of 2, for use building unitary matrices.	
Тор			
See Also			
Reference Util Class Microsoft.Research.Liquid Namespace			

Utilcopyright Property

Copyright notice for Liquid.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Property Value Type: **FSharpFuncUnit**, **Unit**

⊿ See Also

Reference Util Class Microsoft.Research.Liquid Namespace

Utilinteractive Property

[Missing <summary> documentation for "P:Microsoft.Research.Liquid.Util.interactive"]

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

_____ Сору _

static member interactive : bool with get

Property Value Type: Boolean

▲ See Also

Reference Util Class Microsoft.Research.Liquid Namespace

UtiloutputPrefix Property

Global override for the prefix used for output functions (show and variants).

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Property Value Type: String

⊿ See Also

Reference Util Class Microsoft.Research.Liquid Namespace

Utilsqrt2 Property

Highly accurate square root of 2, for use building unitary matrices.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# ______ Copy ______ static member sqrt2 : float with get Property Value Type: Double • See Also Reference Util Class

Microsoft.Research.Liquid Namespace

Util Methods

The Util type exposes the following members.

Methods

	Name	Description
= ∲ S	show	Shows a line of output. This is a command line-callable routine.
≓Ŷ S	getFlgPar	Extracts a boolean macro definition from a command option dictionary.
≓∳ S	getFltPar	Extracts a float macro definition from a command option dictionary.
≓∳ S	getIntPar	Extracts an integer macro definition from a command option dictionary.
⊴∳ S	getIntsPar	Extracts an integer list macro definition from a command option dictionary.
⊴ ≬ S	getStrPar	Extracts a string macro definition from a command option dictionary.
⊴ 0 S	Normal	Add a Normal distribution to the System.Random class. This is an extension method and may

		be used as if it were an instance method on Random.
≓∳ S	PermAry	Create a random permutation array (call .permute with results) This is an extension method and may be used as if it were an instance method on Random.
≡ ≬ S	procStats	Gets current process memory usage statistics.
≡ ≬ S	showa	Shows a line of output to the console and the log file.
≡\$	showBareInd	Shows an indented line of output, with no prefix, to the console and log file. This routine is used by or with various Dump() routines.
= 0 S	showBareLogInd	Shows an indented line of output, with no prefix, to the log file. This routine is used by or with various Dump() routines.
= 0 S	showDump	Dumps out a string with with an optional indentation.
≓ ≬ S	showInd	Shows an indented line of output to the console and log file. This routine is used by or with various Dump() routines.
= ≬ S	showLoga	Shows a line of output to the log file.

≓ ♥ S	showLogInd	Shows an indented line of output to the log file. This routine is used by or with various Dump() routines.
≈≬ S	showProcStats	Print process memory usage statistics to the console and log file.

Тор

⊿ See Also

Reference Util Class Microsoft.Research.Liquid Namespace

Util__show Method

Shows a line of output. This is a command line-callable routine.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Microsoft.Research.Liquid Namespace

UtilgetFlgPar Method

Extracts a boolean macro definition from a command option dictionary.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

dic

Type: System.Collections.GenericDictionaryString, String The dictionary to search

nam

Type: SystemString

The macro name

def

Type: SystemBoolean

The default value to return if the name is not found in the dictionary

Return Value

Type: Boolean

Macro value if found, or else default value

⊿ See Also

Reference Util Class Microsoft.Research.Liquid Namespace

UtilgetFltPar Method

Extracts a float macro definition from a command option dictionary.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

dic

Type: System.Collections.GenericDictionaryString, String The dictionary to search

nam

Type: SystemString

The macro name

def

Type: SystemDouble

The default value to return if the name is not found in the dictionary

Return Value

Type: Double Macro value if found, or else de

Macro value if found, or else default value

J See Also

Reference Util Class Microsoft.Research.Liquid Namespace

UtilgetIntPar Method

Extracts an integer macro definition from a command option dictionary.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

dic

Type: System.Collections.GenericDictionaryString, String The dictionary to search

nam

Type: SystemString

The macro name

def

Type: SystemInt32

The default value to return if the name is not found in the dictionary

Return Value

Type: Int32

Macro value if found, or else default value

⊿ See Also

Reference Util Class Microsoft.Research.Liquid Namespace

UtilgetIntsPar Method

Extracts an integer list macro definition from a command option dictionary.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

dic

Type: System.Collections.GenericDictionaryString, String The dictionary to search

nam

Type: SystemString

The macro name

def

Type: **Microsoft.FSharp.CollectionsFSharpListInt32** The default value to return if the name is not found in the dictionary

Return Value Type: FSharpListInt32 Macro value if found, or else default value

⊿ See Also

Reference Util Class Microsoft.Research.Liquid Namespace

UtilgetStrPar Method

Extracts a string macro definition from a command option dictionary.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

dic

Type: System.Collections.GenericDictionaryString, String The dictionary to search

nam

Type: SystemString

The macro name

def

Type: SystemString

The default value to return if the name is not found in the dictionary

Return Value

Type: String Macro value if found, or else default value

⊿ See Also

Reference Util Class Microsoft.Research.Liquid Namespace
UtilNormal Method

Add a Normal distribution to the System.Random class. This is an extension method and may be used as if it were an instance method on Random.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

Χ

Type: SystemRandom

The instance of System.Random (ignore)

mean

Type: SystemDouble Mean of returned value

sd

Type: SystemDouble Standard deviation of returned value

Return Value Type: Double Random Gaussian value



Reference Util Class Microsoft.Research.Liquid Namespace

UtilPermAry Method

Create a random permutation array (call .permute with results) This is an extension method and may be used as if it were an instance method on Random.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F# _____Copy ____Copy ___COPY ____COPY ___COPY __COPY _COPY __COPY __COPY _COPY __COPY __COPY __COPY _COPY __COPY
```

Parameters

Х

Type: SystemRandom The instance of System.Random (ignore)

len

Type: SystemInt32 Length of array to create

Return Value Type: Int32 Randomized array

J See Also

Reference Util Class Microsoft.Research.Liquid Namespace

UtilprocStats Method

Gets current process memory usage statistics.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F# _____ Copy _____ Static member procStats :

doCollect : bool -> UtilprocStatsT

Parameters

doCollect Type: SystemBoolean Do a garbage collection before reporting?

Return Value Type: UtilprocStatsT procStatsT struct

⊿ See Also

Reference Util Class Microsoft.Research.Liquid Namespace Microsoft.Research.LiquidUtilprocStatsT

Utilshowa Method

Shows a line of output to the console and the log file.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

fmt

Type: Microsoft.FSharp.CorePrintfFormata, Unit, String, Unit printf format parameters

Type Parameters

а

Internal type of the printf string. The F# compiler will deduce this.

Return Value Type: *a* Internal printformat.

⊿ See Also

Reference Util Class Microsoft.Research.Liquid Namespace

UtilshowLoga(PrintfFormata, Unit, String, Unit) Util.logOpen Util.logClose

UtilshowBareInd Method

Shows an indented line of output, with no prefix, to the console and log file. This routine is used by or with various Dump() routines.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

```
static member showBareInd :
    level : int *
    line : string -> unit
```

Parameters

level

Type: SystemInt32

The indentation level. Each level represents a two space indent.

Copy

line

Type: SystemString The text to show.

J See Also

Reference Util Class Microsoft.Research.Liquid Namespace UtilshowInd(Int32, String) UtilshowLogInd(Int32, String) UtilshowBareLogInd(Int32, String)

UtilshowBareLogInd Method

Shows an indented line of output, with no prefix, to the log file. This routine is used by or with various Dump() routines.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

```
static member showBareLogInd :
    level : int *
    line : string -> unit
```

Parameters

level

Type: SystemInt32

The indentation level. Each level represents a two space indent.

Copy

line

Type: SystemString The text to show.

J See Also

Reference Util Class Microsoft.Research.Liquid Namespace UtilshowInd(Int32, String) UtilshowLogInd(Int32, String) UtilshowBareInd(Int32, String)

UtilshowDump Method

Dumps out a string with with an optional indentation.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

Parameters

fO

Type: Microsoft.FSharp.CoreFSharpOptionFSharpFuncInt32, FSharpFuncString, Unit

The optional output function to use. The default is showLogInd.

Ю

Type: Microsoft.FSharp.CoreFSharpOptionInt32

The optional indentation level. The default is 0.

str

Type: SystemString String to output

⊿ See Also

Reference Util Class Microsoft.Research.Liquid Namespace

UtilshowInd Method

Shows an indented line of output to the console and log file. This routine is used by or with various Dump() routines.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

```
static member showInd :
    level : int *
    line : string -> unit
```

Parameters

level

Type: SystemInt32

The indentation level. Each level represents a two space indent.

Copy

line

Type: SystemString The text to show.

J See Also

Reference Util Class Microsoft.Research.Liquid Namespace UtilshowLogInd(Int32, String) UtilshowBareInd(Int32, String) UtilshowBareLogInd(Int32, String)

UtilshowLoga Method

Shows a line of output to the log file.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

fmt

Type: Microsoft.FSharp.CorePrintfFormata, Unit, String, Unit printf format parameters

Type Parameters

а

Internal type of the printf string. The F# compiler will deduce this.

Return Value Type: *a* Internal printformat

⊿ See Also

Reference Util Class Microsoft.Research.Liquid Namespace

Utilshowa(PrintfFormata, Unit, String, Unit) Util.logOpen Util.logClose

UtilshowLogInd Method

Shows an indented line of output to the log file. This routine is used by or with various Dump() routines.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

```
static member showLogInd :
    level : int *
    line : string -> unit
```

Parameters

level

Type: SystemInt32

The indentation level. Each level represents a two space indent.

Copy

line

Type: SystemString The text to show.

J See Also

Reference Util Class Microsoft.Research.Liquid Namespace UtilshowInd(Int32, String) UtilshowBareInd(Int32, String) UtilshowBareLogInd(Int32, String)

UtilshowProcStats Method

Print process memory usage statistics to the console and log file.

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



Parameters

nam

Type: SystemString Tag for this log entry, up to 12 characters

⊿ See Also

Reference Util Class Microsoft.Research.Liquid Namespace

UtilLQDAttribute Class

Allows a function to be visable from a LIQUID script or the command line

▲ Inheritance Hierarchy

```
SystemObject SystemAttribute
Microsoft.Research.LiquidUtilLQDAttribute
```

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



The UtilLQDAttribute type exposes the following members.

Constructors

	Name	Description
= \$	UtilLQDAttribute	Initializes a new instance of the UtilLQDAttribute class

Тор

Properties

	Name	Description
*	Typeld	When implemented in a derived class, gets a unique identifier for this Attribute. (Inherited from Attribute.)

Тор

▲ Methods

	Name	Description
≕ Ø	Equals	Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute.)
≕ Ø	GetHashCode	Returns the hash code for this instance. (Inherited from Attribute.)
= 0	GetType	Gets the Type of the current instance. (Inherited from Object.)
=♥	IsDefaultAttribute	When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from Attribute.)
≡ ∳	Match	When overridden in a derived class, returns a value that indicates whether this instance equals a specified object. (Inherited from Attribute.)

≡0	ToString
Тор	
~ ^	



Reference

Microsoft.Research.Liquid Namespace

Copyright (c) 2015, 2016 Microsoft Corporation (all rights reserved)

Returns a string that

(Inherited from Object.)

represents the current object.

UtilLQDAttribute Constructor

Initializes a new instance of the UtilLQDAttribute class

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax



UtilLQDAttribute Class Microsoft.Research.Liquid Namespace

LQDAttribute Properties

The UtilLQDAttribute type exposes the following members.

▲ Properties

		Name	Description
		TypeId	When implemented in a derived class, gets a unique identifier for this Attribute. (Inherited from Attribute.)
	Тор		
4	See Also	0	
	Reference UtilLQDAttribute Class Microsoft.Research.Liquid Namespace		

LQDAttribute Methods

The UtilLQDAttribute type exposes the following members.

▲ Methods

	Name	Description
= Q	Equals	Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute.)
=♥	GetHashCode	Returns the hash code for this instance. (Inherited from Attribute.)
=∲	GetType	Gets the Type of the current instance. (Inherited from Object.)
≓∳	IsDefaultAttribute	When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from Attribute.)
≡♥	Match	When overridden in a derived class, returns a value that indicates whether this instance equals a specified object. (Inherited from Attribute.)
÷	ToString	Returns a string that

Тор

⊿ See Also

Reference UtilLQDAttribute Class Microsoft.Research.Liquid Namespace

UtilprocStatsT Class

Current process memory usage statistics. Returned by the procStates function.

Inheritance Hierarchy

SystemObject Microsoft.Research.LiquidUtilprocStatsT

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

```
F#
[<SealedAttribute>]
[<SerializableAttribute>]
type procStatsT =
    class
        interface IEquatable<UtilprocStatsT>
        interface IStructuralEquatable
        interface IComparable<UtilprocStatsT>
        interface IComparable
        interface IStructuralComparable
    end
```

The UtilprocStatsT type exposes the following members.

▲ Properties

Name	Description
peakVMMB	Peak virtual memory in megabytes

peakWSMB	Peak working set in megabytes
privMB	Private memory in megabytes
wsetMB	Working set in megabytes

Тор

▲ Methods

	Name	Description
=♥	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
≓ \$	GetHashCode	Serves as the default hash function. (Inherited from Object.)
≓∳	GetType	Gets the Type of the current instance. (Inherited from Object.)
≓∳	ToString	Returns a string that represents the current object. (Inherited from Object.)

Тор

⊿ See Also

Reference Microsoft.Research.Liquid Namespace UtilprocStats(Boolean)

procStatsT Properties

The UtilprocStatsT type exposes the following members.

▲ Properties

	Name	Description
	peakVMMB	Peak virtual memory in megabytes
	peakWSMB	Peak working set in megabytes
*	privMB	Private memory in megabytes
	wsetMB	Working set in megabytes

Тор

⊿ See Also

Reference UtilprocStatsT Class Microsoft.Research.Liquid Namespace

UtilprocStatsTpeakVMMB Property

Peak virtual memory in megabytes

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

member peakVMMB : int with get

Property Value Type: Int32

⊿ See Also

Reference UtilprocStatsT Class Microsoft.Research.Liquid Namespace

UtilprocStatsTpeakWSMB Property

Peak working set in megabytes

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

member peakWSMB : int with get

Property Value Type: Int32

⊿ See Also

Reference UtilprocStatsT Class Microsoft.Research.Liquid Namespace

UtilprocStatsTprivMB Property

Private memory in megabytes

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

▲ Syntax

F#

Сору

member privMB : int with get

Property Value Type: Int32

⊿ See Also

Reference UtilprocStatsT Class Microsoft.Research.Liquid Namespace

UtilprocStatsTwsetMB Property

Working set in megabytes

Namespace: Microsoft.Research.Liquid Assembly: Liquid1 (in Liquid1.dll) Version: 1.0.5981.24943 (1.0.*)

Copy

▲ Syntax

F#

member wsetMB : int with get

Property Value Type: Int32

⊿ See Also

Reference UtilprocStatsT Class Microsoft.Research.Liquid Namespace

procStatsT Methods

The UtilprocStatsT type exposes the following members.

Methods

	Name	Description
≓∳	Equals	Determines whether the specified object is equal to the current object. (Inherited from Object.)
=♥	GetHashCode	Serves as the default hash function. (Inherited from Object.)
=♥	GetType	Gets the Type of the current instance. (Inherited from Object.)
=♥	ToString	Returns a string that represents the current object. (Inherited from Object.)

Тор

⊿ See Also

Reference UtilprocStatsT Class Microsoft.Research.Liquid Namespace
