# Quantum++

v1.2

Generated by Doxygen 1.8.14

# Contents


**1 Quantum++** **1**

**2 Namespace Index** **3**

2.1 Namespace List . . . . . 3

**3 Hierarchical Index** **5**

3.1 Class Hierarchy . . . . . 5

**4 Class Index** **7**

4.1 Class List . . . . . 7

**5 File Index** **11**

5.1 File List . . . . . 11

**6 Namespace Documentation** **13**

6.1 qpp Namespace Reference . . . . . 13

6.1.1 Detailed Description . . . . . 26

6.1.2 Typedef Documentation . . . . . 26

6.1.2.1 bigint . . . . . 26

6.1.2.2 bra . . . . . 27

6.1.2.3 cmat . . . . . 27

6.1.2.4 cplx . . . . . 27

6.1.2.5 dmat . . . . . 27

6.1.2.6 dyn_col_vect . . . . . 27

6.1.2.7 dyn_mat . . . . . 28

6.1.2.8 dyn_row_vect . . . . . 28

6.1.2.9 idx . . . . . . 28
6.1.2.10 ket . . . . . . 28
6.1.2.11 to_void . . . . . . 29
6.1.3 Function Documentation . . . . . . 29
6.1.3.1 absm() . . . . . . 29
6.1.3.2 abssq() [1/3] . . . . . . 29
6.1.3.3 abssq() [2/3] . . . . . . 30
6.1.3.4 abssq() [3/3] . . . . . . 30
6.1.3.5 adjoint() . . . . . . 30
6.1.3.6 anticomm() . . . . . . 31
6.1.3.7 apply() [1/5] . . . . . . 31
6.1.3.8 apply() [2/5] . . . . . . 32
6.1.3.9 apply() [3/5] . . . . . . 32
6.1.3.10 apply() [4/5] . . . . . . 33
6.1.3.11 apply() [5/5] . . . . . . 33
6.1.3.12 applyCTRL() [1/2] . . . . . . 34
6.1.3.13 applyCTRL() [2/2] . . . . . . 35
6.1.3.14 applyQFT() . . . . . . 35
6.1.3.15 applyTFQ() . . . . . . 36
6.1.3.16 avg() . . . . . . 36
6.1.3.17 bloch2rho() . . . . . . 37
6.1.3.18 choi2kraus() . . . . . . 37
6.1.3.19 choi2super() . . . . . . 38
6.1.3.20 comm() . . . . . . 38
6.1.3.21 complement() . . . . . . 39
6.1.3.22 compperm() . . . . . . 39
6.1.3.23 concurrence() . . . . . . 39
6.1.3.24 conjugate() . . . . . . 40
6.1.3.25 contfrac2x() . . . . . . 40
6.1.3.26 convergents() [1/2] . . . . . . 41

6.1.3.27 convergents() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 41
6.1.3.28 cor() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42
6.1.3.29 cosm() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42
6.1.3.30 cov() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42
6.1.3.31 cwise() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
6.1.3.32 det() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
6.1.3.33 dirsum() [1/4] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 44
6.1.3.34 dirsum() [2/4] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 44
6.1.3.35 dirsum() [3/4] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 45
6.1.3.36 dirsum() [4/4] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 45
6.1.3.37 dirsumpow() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 45
6.1.3.38 disp() [1/5] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 46
6.1.3.39 disp() [2/5] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 46
6.1.3.40 disp() [3/5] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47
6.1.3.41 disp() [4/5] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47
6.1.3.42 disp() [5/5] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48
6.1.3.43 egcd() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48
6.1.3.44 eig() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 49
6.1.3.45 entanglement() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 49
6.1.3.46 entanglement() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 50
6.1.3.47 entropy() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 50
6.1.3.48 entropy() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51
6.1.3.49 evals() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51
6.1.3.50 evects() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51
6.1.3.51 expm() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 52
6.1.3.52 factors() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 52
6.1.3.53 funm() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53
6.1.3.54 gcd() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53
6.1.3.55 gcd() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 54
6.1.3.56 gconcurrence() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 54

6.1.3.57 grams() [1/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 55
6.1.3.58 grams() [2/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 55
6.1.3.59 grams() [3/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 55
6.1.3.60 hash_eigen() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 56
6.1.3.61 heig() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 56
6.1.3.62 hevals() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57
6.1.3.63 hevects() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57
6.1.3.64 inverse() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 58
6.1.3.65 invperm() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 58
6.1.3.66 ip() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 58
6.1.3.67 ip() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 59
6.1.3.68 isprime() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 59
6.1.3.69 kraus2choi() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60
6.1.3.70 kraus2super() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60
6.1.3.71 kron() [1/4] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 61
6.1.3.72 kron() [2/4] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 61
6.1.3.73 kron() [3/4] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 62
6.1.3.74 kron() [4/4] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 62
6.1.3.75 kronpow() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 62
6.1.3.76 lcm() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 63
6.1.3.77 lcm() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 63
6.1.3.78 load() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 64
6.1.3.79 loadMATLAB() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 64
6.1.3.80 loadMATLAB() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 65
6.1.3.81 logdet() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 66
6.1.3.82 logm() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 66
6.1.3.83 lognegativity() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 67
6.1.3.84 lognegativity() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 67
6.1.3.85 marginalX() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 67
6.1.3.86 marginalY() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 69

6.1.3.87 measure() [1/9] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 69
6.1.3.88 measure() [2/9] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 70
6.1.3.89 measure() [3/9] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 70
6.1.3.90 measure() [4/9] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 70
6.1.3.91 measure() [5/9] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 71
6.1.3.92 measure() [6/9] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 72
6.1.3.93 measure() [7/9] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 72
6.1.3.94 measure() [8/9] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 73
6.1.3.95 measure() [9/9] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 74
6.1.3.96 measure_seq() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . 74
6.1.3.97 measure_seq() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . 75
6.1.3.98 mket() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 75
6.1.3.99 mket() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 76
6.1.3.100 modinv() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 76
6.1.3.101 modmul() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77
6.1.3.102 modpow() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77
6.1.3.103 mprj() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 78
6.1.3.104 mprj() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 79
6.1.3.105 multiidx2n() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 79
6.1.3.106 n2multiidx() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 80
6.1.3.107 negativity() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 80
6.1.3.108 negativity() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 80
6.1.3.109 norm() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 81
6.1.3.110 normalize() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 81
6.1.3.111 omega() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 82
6.1.3.112 operator"""" _i() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 82
6.1.3.113 powm() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 82
6.1.3.114 prj() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 83
6.1.3.115 prod() [1/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 83
6.1.3.116 prod() [2/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 84

6.1.3.117 prod() [3/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 84
6.1.3.118 ptrace() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 84
6.1.3.119 ptrace() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 85
6.1.3.120 ptrace1() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 85
6.1.3.121 ptrace1() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 86
6.1.3.122 ptrace2() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 86
6.1.3.123 ptrace2() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 87
6.1.3.124 ptranspose() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 87
6.1.3.125 ptranspose() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 88
6.1.3.126 QFT() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 88
6.1.3.127 qmutualinfo() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 89
6.1.3.128 qmutualinfo() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 89
6.1.3.129 rand() [1/5] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 90
6.1.3.130 rand() [2/5] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 90
6.1.3.131 rand() [3/5] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 91
6.1.3.132 rand() [4/5] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 91
6.1.3.133 rand() [5/5] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 92
6.1.3.134 randH() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 92
6.1.3.135 randidx() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 93
6.1.3.136 randket() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 93
6.1.3.137 randkraus() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 93
6.1.3.138 randn() [1/4] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 94
6.1.3.139 randn() [2/4] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 94
6.1.3.140 randn() [3/4] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 95
6.1.3.141 randn() [4/4] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 95
6.1.3.142 randperm() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 96
6.1.3.143 randprime() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 96
6.1.3.144 randprob() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 97
6.1.3.145 randrho() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 97
6.1.3.146 randU() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 97

6.1.3.147 randV() . . . 98
6.1.3.148 renyi() [1/2] . . . 98
6.1.3.149 renyi() [2/2] . . . 99
6.1.3.150 reshape() . . . 99
6.1.3.151 rho2bloch() . . . 100
6.1.3.152 rho2pure() . . . 100
6.1.3.153 save() . . . 101
6.1.3.154 saveMATLAB() [1/2] . . . 101
6.1.3.155 saveMATLAB() [2/2] . . . 102
6.1.3.156 schatten() . . . 102
6.1.3.157 schmidtA() [1/2] . . . 103
6.1.3.158 schmidtA() [2/2] . . . 103
6.1.3.159 schmidtB() [1/2] . . . 103
6.1.3.160 schmidtB() [2/2] . . . 104
6.1.3.161 schmidtcoeffs() [1/2] . . . 104
6.1.3.162 schmidtcoeffs() [2/2] . . . 105
6.1.3.163 schmidtprobs() [1/2] . . . 105
6.1.3.164 schmidtprobs() [2/2] . . . 106
6.1.3.165 sigma() . . . 106
6.1.3.166 sinm() . . . 107
6.1.3.167 spectralpowm() . . . 107
6.1.3.168 sqrtm() . . . 108
6.1.3.169 sum() [1/3] . . . 108
6.1.3.170 sum() [2/3] . . . 108
6.1.3.171 sum() [3/3] . . . 109
6.1.3.172 super2choi() . . . 109
6.1.3.173 svals() . . . 110
6.1.3.174 svd() . . . 110
6.1.3.175 svdU() . . . 110
6.1.3.176 svdV() . . . 111

6.1.3.177 syspermute() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 111
6.1.3.178 syspermute() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 112
6.1.3.179 TFQ() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 112
6.1.3.180 trace() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 113
6.1.3.181 transpose() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 113
6.1.3.182 tsallis() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 113
6.1.3.183 tsallis() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 114
6.1.3.184 uniform() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 114
6.1.3.185 var() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 115
6.1.3.186 x2contfrac() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 115
6.1.4 Variable Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 116
6.1.4.1 chop . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 116
6.1.4.2 ee . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 116
6.1.4.3 infty . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 116
6.1.4.4 maxn . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 116
6.1.4.5 pi . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 116
6.2 qpp::exception Namespace Reference . . . . . . . . . . . . . . . . . . . . . . . . . 116
6.2.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 118
6.3 qpp::experimental Namespace Reference . . . . . . . . . . . . . . . . . . . . . . . 118
6.3.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 118
6.4 qpp::internal Namespace Reference . . . . . . . . . . . . . . . . . . . . . . . . . . 118
6.4.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 120
6.4.2 Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 120
6.4.2.1 check_cvector() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 120
6.4.2.2 check_dims() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 120
6.4.2.3 check_dims_match_cvect() . . . . . . . . . . . . . . . . . . . . . . . . . . 120
6.4.2.4 check_dims_match_mat() . . . . . . . . . . . . . . . . . . . . . . . . . . . 120
6.4.2.5 check_dims_match_rvect() . . . . . . . . . . . . . . . . . . . . . . . . . . 121
6.4.2.6 check_eq_dims() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 121
6.4.2.7 check_matching_sizes() . . . . . . . . . . . . . . . . . . . . . . . . . . . . 121

6.4.2.8 check_no_duplicates() . . . 121
6.4.2.9 check_nonzero_size() . . . 121
6.4.2.10 check_perm() . . . 121
6.4.2.11 check_qubit_cvector() . . . 122
6.4.2.12 check_qubit_matrix() . . . 122
6.4.2.13 check_qubit_rvector() . . . 122
6.4.2.14 check_qubit_vector() . . . 122
6.4.2.15 check_rvector() . . . 122
6.4.2.16 check_square_mat() . . . 122
6.4.2.17 check_subsys_match_dims() . . . 123
6.4.2.18 check_vector() . . . 123
6.4.2.19 dirsum2() . . . 123
6.4.2.20 get_dim_subsys() . . . 123
6.4.2.21 get_num_subsys() . . . 123
6.4.2.22 hash_combine() . . . 123
6.4.2.23 kron2() . . . 124
6.4.2.24 multiidx2n() . . . 124
6.4.2.25 n2multiidx() . . . 124
6.4.2.26 variadic_vector_emplace() [1/2] . . . 124
6.4.2.27 variadic_vector_emplace() [2/2] . . . 124
6.5 qpp::literals Namespace Reference . . . 125
6.5.1 Function Documentation . . . 125
6.5.1.1 operator"""" _bra() . . . 125
6.5.1.2 operator"""" _i() . . . 125
6.5.1.3 operator"""" _ket() . . . 126
6.5.1.4 operator"""" _prj() . . . 126

7 Class Documentation 129
7.1 qpp::Bit_circuit Class Reference . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 129
7.1.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 131
7.1.2 Constructor & Destructor Documentation . . . . . . . . . . . . . . . . . . . . . . . . 131
7.1.2.1 Bit_circuit() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 131
7.1.2.2 Bit_circuit() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 131
7.1.2.3 ∼Bit_circuit() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 132
7.1.3 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . 132
7.1.3.1 CNOT() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 132
7.1.3.2 FRED() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 132
7.1.3.3 get_gate_count() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 133
7.1.3.4 get_gate_depth() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 133
7.1.3.5 NOT() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 134
7.1.3.6 reset() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 134
7.1.3.7 SWAP() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 134
7.1.3.8 TOF() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 135
7.1.3.9 X() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 135
7.1.4 Member Data Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 136
7.1.4.1 bCNOT_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 136
7.1.4.2 bFRED_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 136
7.1.4.3 bNOT_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 136
7.1.4.4 bSWAP_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 136
7.1.4.5 bTOF_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 136
7.1.4.6 btotal_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 136
7.1.4.7 count_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 137
7.1.4.8 depth_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 137
7.2 qpp::Codes Class Reference . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 137
7.2.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 138
7.2.2 Member Enumeration Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . 138
7.2.2.1 Type . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 138

7.2.3 Constructor & Destructor Documentation . . . 139
7.2.3.1 Codes() . . . 139
7.2.3.2 ∼Codes() . . . 139
7.2.4 Member Function Documentation . . . 139
7.2.4.1 codeword() . . . 139
7.2.5 Friends And Related Function Documentation . . . 139
7.2.5.1 internal::Singleton< const Codes > . . . 140
7.3 qpp::exception::CustomException Class Reference . . . 140
7.3.1 Detailed Description . . . 141
7.3.2 Constructor & Destructor Documentation . . . 141
7.3.2.1 CustomException() . . . 142
7.3.3 Member Function Documentation . . . 142
7.3.3.1 type_description() . . . 142
7.3.4 Member Data Documentation . . . 142
7.3.4.1 what_ . . . 142
7.4 qpp::exception::DimsInvalid Class Reference . . . 143
7.4.1 Detailed Description . . . 144
7.4.2 Member Function Documentation . . . 144
7.4.2.1 Exception() . . . 144
7.4.2.2 type_description() . . . 144
7.5 qpp::exception::DimsMismatchCvector Class Reference . . . 145
7.5.1 Detailed Description . . . 146
7.5.2 Member Function Documentation . . . 146
7.5.2.1 Exception() . . . 146
7.5.2.2 type_description() . . . 146
7.6 qpp::exception::DimsMismatchMatrix Class Reference . . . 147
7.6.1 Detailed Description . . . 148
7.6.2 Member Function Documentation . . . 148
7.6.2.1 Exception() . . . 148
7.6.2.2 type_description() . . . 148

7.7 qpp::exception::DimsMismatchRvector Class Reference . . . . . . . . . . . . . . . . . . . . . . 149
7.7.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 150
7.7.2 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . 150
7.7.2.1 Exception() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 150
7.7.2.2 type_description() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 150
7.8 qpp::exception::DimsMismatchVector Class Reference . . . . . . . . . . . . . . . . . . . . . . 151
7.8.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 152
7.8.2 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . 152
7.8.2.1 Exception() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 152
7.8.2.2 type_description() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 152
7.9 qpp::exception::DimsNotEqual Class Reference . . . . . . . . . . . . . . . . . . . . . . . . . . 153
7.9.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 154
7.9.2 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . 154
7.9.2.1 Exception() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 154
7.9.2.2 type_description() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 154
7.10 qpp::internal::Display_Impl_ Struct Reference . . . . . . . . . . . . . . . . . . . . . . . . . . 155
7.10.1 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . 155
7.10.1.1 display_impl_() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 155
7.11 qpp::exception::Duplicates Class Reference . . . . . . . . . . . . . . . . . . . . . . . . . . . 156
7.11.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 157
7.11.2 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . 157
7.11.2.1 Exception() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 157
7.11.2.2 type_description() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 157
7.12 qpp::Dynamic_bitset Class Reference . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 158
7.12.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 160
7.12.2 Member Typedef Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . 160
7.12.2.1 storage_type . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 160
7.12.2.2 value_type . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 160
7.12.3 Constructor & Destructor Documentation . . . . . . . . . . . . . . . . . . . . . . . 160
7.12.3.1 Dynamic_bitset() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 160

7.12.3.2 ∼Dynamic_bitset() . . . . . . . . . . . . . . . . . . . . . . . . . . . . 161
7.12.4 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . 161
7.12.4.1 all() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 161
7.12.4.2 any() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 161
7.12.4.3 count() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 161
7.12.4.4 data() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 162
7.12.4.5 display() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 162
7.12.4.6 flip() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 162
7.12.4.7 flip() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 163
7.12.4.8 get() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 163
7.12.4.9 index_() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 163
7.12.4.10 none() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 164
7.12.4.11 offset_() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 164
7.12.4.12 operator"!=() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 164
7.12.4.13 operator-() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 165
7.12.4.14 operator==() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 165
7.12.4.15 rand() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 165
7.12.4.16 rand() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 166
7.12.4.17 reset() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 166
7.12.4.18 reset() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 166
7.12.4.19 set() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 167
7.12.4.20 set() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 167
7.12.4.21 size() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 167
7.12.4.22 storage_size() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 167
7.12.4.23 to_string() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 168
7.12.5 Member Data Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 168
7.12.5.1 n_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 168
7.12.5.2 storage_size_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 168
7.12.5.3 v_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 169
7.13 qpp::internal::EqualEigen Class Reference . . . . . . . . . . . . . . . . . . . . . . . . . . 169

7.13.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 169
7.13.2 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 169
7.13.2.1 operator()() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 169
7.14 qpp::exception::Exception Class Reference . . . . . . . . . . . . . . . . . . . . . . . . . . . . 170
7.14.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 171
7.14.2 Constructor & Destructor Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . 172
7.14.2.1 Exception() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 172
7.14.3 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 172
7.14.3.1 type_description() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 172
7.14.3.2 what() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 173
7.14.4 Member Data Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 173
7.14.4.1 msg_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 173
7.14.4.2 where_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 173
7.15 qpp::Gates Class Reference . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 173
7.15.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 176
7.15.2 Constructor & Destructor Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . 176
7.15.2.1 Gates() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 176
7.15.2.2 ∼Gates() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 176
7.15.3 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 176
7.15.3.1 CTRL() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 176
7.15.3.2 expandout() [1/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 177
7.15.3.3 expandout() [2/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 177
7.15.3.4 expandout() [3/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 178
7.15.3.5 Fd() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 179
7.15.3.6 get_name() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 179
7.15.3.7 Id() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 180
7.15.3.8 MODMUL() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 180
7.15.3.9 Rn() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 181
7.15.3.10 RX() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 181
7.15.3.11 RY() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 181

7.15.3.12 RZ() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 182
7.15.3.13 SWAPd() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 182
7.15.3.14 Xd() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 182
7.15.3.15 Zd() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 183
7.15.4 Friends And Related Function Documentation . . . . . . . . . . . . . . . . . . . . . . 183
7.15.4.1 internal::Singleton< const Gates > . . . . . . . . . . . . . . . . . . . . . . . 183
7.15.5 Member Data Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . 183
7.15.5.1 CNOT . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 183
7.15.5.2 CNOTba . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 184
7.15.5.3 CZ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 184
7.15.5.4 FRED . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 184
7.15.5.5 H . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 184
7.15.5.6 Id2 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 184
7.15.5.7 S . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 184
7.15.5.8 SWAP . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 185
7.15.5.9 T . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 185
7.15.5.10 TOF . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 185
7.15.5.11 X . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 185
7.15.5.12 Y . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 185
7.15.5.13 Z . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 185
7.16 qpp::QCircuit::GateStep Struct Reference . . . . . . . . . . . . . . . . . . . . . . . . . . 186
7.16.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 187
7.16.2 Constructor & Destructor Documentation . . . . . . . . . . . . . . . . . . . . . . . . 187
7.16.2.1 GateStep() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 187
7.16.2.2 GateStep() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 187
7.16.3 Member Data Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . 187
7.16.3.1 ctrl_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 187
7.16.3.2 gate_hash_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 188
7.16.3.3 gate_type_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 188
7.16.3.4 name_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 188

7.16.3.5 target_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 188
7.17 qpp::internal::HashEigen Class Reference . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 188
7.17.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 189
7.17.2 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 189
7.17.2.1 operator()() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 189
7.18 qpp::IDisplay Class Reference . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 189
7.18.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 190
7.18.2 Constructor & Destructor Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . 190
7.18.2.1 IDisplay() [1/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 190
7.18.2.2 IDisplay() [2/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 191
7.18.2.3 IDisplay() [3/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 191
7.18.2.4 ∼IDisplay() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 191
7.18.3 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 191
7.18.3.1 display() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 191
7.18.3.2 operator=() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 191
7.18.3.3 operator=() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 192
7.18.4 Friends And Related Function Documentation . . . . . . . . . . . . . . . . . . . . . . . 192
7.18.4.1 operator<< . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 192
7.19 qpp::IJSON Class Reference . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 192
7.19.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 193
7.19.2 Constructor & Destructor Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . 193
7.19.2.1 IJSON() [1/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 193
7.19.2.2 IJSON() [2/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 193
7.19.2.3 IJSON() [3/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 193
7.19.2.4 ∼IJSON() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 194
7.19.3 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 194
7.19.3.1 operator=() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 194
7.19.3.2 operator=() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 194
7.19.3.3 to_JSON() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 194
7.20 qpp::Init Class Reference . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 195

7.20.1 Detailed Description . . . . . 196
7.20.2 Constructor & Destructor Documentation . . . . . 196
7.20.2.1 Init() . . . . . 196
7.20.2.2 ∼Init() . . . . . 196
7.20.3 Friends And Related Function Documentation . . . . . 196
7.20.3.1 internal::Singleton< const Init > . . . . . 196
7.21 qpp::exception::InvalidIterator Class Reference . . . . . 197
7.21.1 Detailed Description . . . . . 198
7.21.2 Member Function Documentation . . . . . 198
7.21.2.1 Exception() . . . . . 198
7.21.2.2 type_description() . . . . . 198
7.22 qpp::internal::IOManipEigen Class Reference . . . . . 199
7.22.1 Constructor & Destructor Documentation . . . . . 200
7.22.1.1 IOManipEigen() [1/2] . . . . . 200
7.22.1.2 IOManipEigen() [2/2] . . . . . 200
7.22.2 Member Function Documentation . . . . . 200
7.22.2.1 display() . . . . . 200
7.22.3 Member Data Documentation . . . . . 200
7.22.3.1 A_ . . . . . 201
7.22.3.2 chop_ . . . . . 201
7.23 qpp::internal::IOManipPointer< PointerType > Class Template Reference . . . . . 201
7.23.1 Constructor & Destructor Documentation . . . . . 202
7.23.1.1 IOManipPointer() [1/2] . . . . . 203
7.23.1.2 IOManipPointer() [2/2] . . . . . 203
7.23.2 Member Function Documentation . . . . . 203
7.23.2.1 display() . . . . . 203
7.23.2.2 operator=() . . . . . 203
7.23.3 Member Data Documentation . . . . . 203
7.23.3.1 end_ . . . . . 204
7.23.3.2 N_ . . . . . 204

7.23.3.3 p_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 204
7.23.3.4 separator_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 204
7.23.3.5 start_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 204
7.24 qpp::internal::IOManipRange< InputIterator > Class Template Reference . . . . . . . . . . . . . . 205
7.24.1 Constructor & Destructor Documentation . . . . . . . . . . . . . . . . . . . . . . . . 206
7.24.1.1 IOManipRange() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . 206
7.24.1.2 IOManipRange() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . 206
7.24.2 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . 206
7.24.2.1 display() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 206
7.24.2.2 operator=() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 207
7.24.3 Member Data Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 207
7.24.3.1 end_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 207
7.24.3.2 first_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 207
7.24.3.3 last_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 207
7.24.3.4 separator_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 207
7.24.3.5 start_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 207
7.25 qpp::is_complex< T > Struct Template Reference . . . . . . . . . . . . . . . . . . . . . . . . 208
7.25.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 208
7.26 qpp::is_complex< std::complex< T > > Struct Template Reference . . . . . . . . . . . . . . . . 209
7.26.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 209
7.27 qpp::is_iterable< T, typename > Struct Template Reference . . . . . . . . . . . . . . . . . . . . 210
7.27.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 210
7.28 qpp::is_iterable< T, to_void< decltype(std::declval< T >().begin()), decltype(std::declval< T >().end()), decltype(*(std::declval< T >().begin()))> > Struct Template Reference . . . . . . . . 211
7.28.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 212
7.29 qpp::is_matrix_expression< Derived > Struct Template Reference . . . . . . . . . . . . . . . . . 212
7.29.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 212
7.30 qpp::QCircuit::iterator Class Reference . . . . . . . . . . . . . . . . . . . . . . . . . . . . 213
7.30.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 214
7.30.2 Member Typedef Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . 214
7.30.2.1 difference_type . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 214

7.30.2.2 iterator_category . . . . . . . . . . . . . . . . . . . . . . . . . . . . 214
7.30.2.3 pointer . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 215
7.30.2.4 reference . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 215
7.30.2.5 value_type . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 215
7.30.3 Constructor & Destructor Documentation . . . . . . . . . . . . . . . . . . . . 215
7.30.3.1 iterator() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 215
7.30.3.2 iterator() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 215
7.30.4 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . 215
7.30.4.1 operator"!=() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 215
7.30.4.2 operator*() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 216
7.30.4.3 operator++() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . 216
7.30.4.4 operator++() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . 216
7.30.4.5 operator=() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 217
7.30.4.6 operator==() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 217
7.30.4.7 set_begin_() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 217
7.30.4.8 set_end_() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 217
7.30.5 Member Data Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . 218
7.30.5.1 elem_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 218
7.30.5.2 qc_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 218
7.31 qpp::make_void< Ts > Struct Template Reference . . . . . . . . . . . . . . . . . . . 218
7.31.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 218
7.31.2 Member Typedef Documentation . . . . . . . . . . . . . . . . . . . . . . . . . 219
7.31.2.1 type . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 219
7.32 qpp::exception::MatrixMismatchSubsys Class Reference . . . . . . . . . . . . . . . . . 219
7.32.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 220
7.32.2 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . 220
7.32.2.1 Exception() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 220
7.32.2.2 type_description() . . . . . . . . . . . . . . . . . . . . . . . . . . . . 221
7.33 qpp::exception::MatrixNotCvector Class Reference . . . . . . . . . . . . . . . . . . . 221
7.33.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 222

7.33.2 Member Function Documentation . . . 222
7.33.2.1 Exception() . . . 222
7.33.2.2 type_description() . . . 223
7.34 qpp::exception::MatrixNotRvector Class Reference . . . 223
7.34.1 Detailed Description . . . 224
7.34.2 Member Function Documentation . . . 224
7.34.2.1 Exception() . . . 224
7.34.2.2 type_description() . . . 225
7.35 qpp::exception::MatrixNotSquare Class Reference . . . 225
7.35.1 Detailed Description . . . 226
7.35.2 Member Function Documentation . . . 226
7.35.2.1 Exception() . . . 226
7.35.2.2 type_description() . . . 227
7.36 qpp::exception::MatrixNotSquareNorCvector Class Reference . . . 227
7.36.1 Detailed Description . . . 228
7.36.2 Member Function Documentation . . . 228
7.36.2.1 Exception() . . . 228
7.36.2.2 type_description() . . . 229
7.37 qpp::exception::MatrixNotSquareNorRvector Class Reference . . . 229
7.37.1 Detailed Description . . . 230
7.37.2 Member Function Documentation . . . 230
7.37.2.1 Exception() . . . 230
7.37.2.2 type_description() . . . 231
7.38 qpp::exception::MatrixNotSquareNorVector Class Reference . . . 231
7.38.1 Detailed Description . . . 232
7.38.2 Member Function Documentation . . . 232
7.38.2.1 Exception() . . . 232
7.38.2.2 type_description() . . . 233
7.39 qpp::exception::MatrixNotVector Class Reference . . . 233
7.39.1 Detailed Description . . . 234

7.39.2 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . 234
7.39.2.1 Exception() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 234
7.39.2.2 type_description() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 235
7.40 qpp::QCircuit::MeasureStep Struct Reference . . . . . . . . . . . . . . . . . . . . . . . . . . . 235
7.40.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 236
7.40.2 Constructor & Destructor Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . 236
7.40.2.1 MeasureStep() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 236
7.40.2.2 MeasureStep() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 236
7.40.3 Member Data Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 237
7.40.3.1 c_reg_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 237
7.40.3.2 mats_hash_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 237
7.40.3.3 measurement_type_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 237
7.40.3.4 name_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 237
7.40.3.5 target_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 237
7.41 qpp::exception::NoCodeword Class Reference . . . . . . . . . . . . . . . . . . . . . . . . . . 238
7.41.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 239
7.41.2 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 239
7.41.2.1 Exception() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 239
7.41.2.2 type_description() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 239
7.42 qpp::NoiseBase< T > Class Template Reference . . . . . . . . . . . . . . . . . . . . . . . . . 240
7.42.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 241
7.42.2 Member Typedef Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 241
7.42.2.1 noise_type . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 241
7.42.3 Constructor & Destructor Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . 242
7.42.3.1 NoiseBase() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 242
7.42.3.2 NoiseBase() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 242
7.42.3.3 ∼NoiseBase() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 243
7.42.4 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 243
7.42.4.1 compute_probs_() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 243
7.42.4.2 compute_state_() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 243

7.42.4.3 get_d() . . . . . . 244
7.42.4.4 get_Ks() . . . . . . 244
7.42.4.5 get_last_idx() . . . . . . 244
7.42.4.6 get_last_K() . . . . . . 244
7.42.4.7 get_last_p() . . . . . . 245
7.42.4.8 get_probs() . . . . . . 245
7.42.4.9 operator()() [1/3] . . . . . . 245
7.42.4.10 operator()() [2/3] . . . . . . 245
7.42.4.11 operator()() [3/3] . . . . . . 246
7.42.5 Member Data Documentation . . . . . . 246
7.42.5.1 d_ . . . . . . 246
7.42.5.2 generated_ . . . . . . 247
7.42.5.3 i_ . . . . . . 247
7.42.5.4 Ks_ . . . . . . 247
7.42.5.5 probs_ . . . . . . 247
7.43 qpp::NoiseType Class Reference . . . . . . 247
7.43.1 Detailed Description . . . . . . 248
7.44 qpp::exception::NotBipartite Class Reference . . . . . . 248
7.44.1 Detailed Description . . . . . . 249
7.44.2 Member Function Documentation . . . . . . 249
7.44.2.1 Exception() . . . . . . 249
7.44.2.2 type_description() . . . . . . 250
7.45 qpp::exception::NotImplemented Class Reference . . . . . . 250
7.45.1 Detailed Description . . . . . . 251
7.45.2 Member Function Documentation . . . . . . 251
7.45.2.1 Exception() . . . . . . 251
7.45.2.2 type_description() . . . . . . 252
7.46 qpp::exception::NotQubitCvector Class Reference . . . . . . 252
7.46.1 Detailed Description . . . . . . 253
7.46.2 Member Function Documentation . . . . . . 253

7.46.2.1 Exception() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 253
7.46.2.2 type_description() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 254
7.47 qpp::exception::NotQubitMatrix Class Reference . . . . . . . . . . . . . . . . . . . . . . . . 254
7.47.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 255
7.47.2 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . 255
7.47.2.1 Exception() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 255
7.47.2.2 type_description() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 256
7.48 qpp::exception::NotQubitRvector Class Reference . . . . . . . . . . . . . . . . . . . . . . . 256
7.48.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 257
7.48.2 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . 257
7.48.2.1 Exception() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 257
7.48.2.2 type_description() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 258
7.49 qpp::exception::NotQubitSubsys Class Reference . . . . . . . . . . . . . . . . . . . . . . . 258
7.49.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 259
7.49.2 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . 259
7.49.2.1 Exception() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 259
7.49.2.2 type_description() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 260
7.50 qpp::exception::NotQubitVector Class Reference . . . . . . . . . . . . . . . . . . . . . . . . 260
7.50.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 261
7.50.2 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . 261
7.50.2.1 Exception() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 261
7.50.2.2 type_description() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 262
7.51 qpp::exception::OutOfRange Class Reference . . . . . . . . . . . . . . . . . . . . . . . . . 262
7.51.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 263
7.51.2 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . 263
7.51.2.1 Exception() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 263
7.51.2.2 type_description() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 264
7.52 qpp::exception::PermInvalid Class Reference . . . . . . . . . . . . . . . . . . . . . . . . . 264
7.52.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 265
7.52.2 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . 265

7.52.2.1 Exception() . . . . . 265
7.52.2.2 type_description() . . . . . 266
7.53 qpp::exception::PermMismatchDims Class Reference . . . . . 266
7.53.1 Detailed Description . . . . . 267
7.53.2 Member Function Documentation . . . . . 267
7.53.2.1 Exception() . . . . . 267
7.53.2.2 type_description() . . . . . 268
7.54 qpp::QCircuit Class Reference . . . . . 268
7.54.1 Detailed Description . . . . . 273
7.54.2 Member Typedef Documentation . . . . . 273
7.54.2.1 const_iterator . . . . . 273
7.54.3 Member Enumeration Documentation . . . . . 273
7.54.3.1 GateType . . . . . 273
7.54.3.2 MeasureType . . . . . 274
7.54.3.3 StepType . . . . . 274
7.54.4 Constructor & Destructor Documentation . . . . . 274
7.54.4.1 QCircuit() . . . . . 274
7.54.4.2 ∼QCircuit() . . . . . 275
7.54.5 Member Function Documentation . . . . . 275
7.54.5.1 add_hash_() . . . . . 275
7.54.5.2 begin() [1/2] . . . . . 275
7.54.5.3 begin() [2/2] . . . . . 276
7.54.5.4 cbegin() . . . . . 276
7.54.5.5 cCTRL() [1/4] . . . . . 276
7.54.5.6 cCTRL() [2/4] . . . . . 277
7.54.5.7 cCTRL() [3/4] . . . . . 277
7.54.5.8 cCTRL() [4/4] . . . . . 277
7.54.5.9 cCTRL_custom() . . . . . 278
7.54.5.10 cend() . . . . . 278
7.54.5.11 CTRL() [1/4] . . . . . 279

7.54.5.12 CTRL() [2/4] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 279
7.54.5.13 CTRL() [3/4] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 280
7.54.5.14 CTRL() [4/4] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 280
7.54.5.15 CTRL_custom() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 281
7.54.5.16 display() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 281
7.54.5.17 end() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 282
7.54.5.18 end() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 282
7.54.5.19 gate() [1/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 282
7.54.5.20 gate() [2/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 282
7.54.5.21 gate() [3/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 283
7.54.5.22 gate_custom() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 283
7.54.5.23 gate_fan() [1/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 284
7.54.5.24 gate_fan() [2/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 284
7.54.5.25 gate_fan() [3/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 285
7.54.5.26 get_cmat_hash_tbl_() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 285
7.54.5.27 get_d() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 285
7.54.5.28 get_gate_count() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 286
7.54.5.29 get_gate_depth() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 286
7.54.5.30 get_gates_() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 286
7.54.5.31 get_measured() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 287
7.54.5.32 get_measured() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 287
7.54.5.33 get_measurement_count() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . 287
7.54.5.34 get_measurement_count() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . 287
7.54.5.35 get_measurements_() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 288
7.54.5.36 get_name() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 288
7.54.5.37 get_nc() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 288
7.54.5.38 get_non_measured() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 289
7.54.5.39 get_nop_count() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 289
7.54.5.40 get_nq() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 289
7.54.5.41 get_step_count() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 289

7.54.5.42 measureV() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 289
7.54.5.43 measureV() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 290
7.54.5.44 measureZ() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 290
7.54.5.45 nop() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 291
7.54.5.46 QFT() [1/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 291
7.54.5.47 QFT() [2/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 291
7.54.5.48 QFT() [3/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 292
7.54.5.49 TFQ() [1/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 292
7.54.5.50 TFQ() [2/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 293
7.54.5.51 TFQ() [3/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 293
7.54.5.52 to_JSON() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 293
7.54.6 Friends And Related Function Documentation . . . . . . . . . . . . . . . . . . . . . 294
7.54.6.1 operator<< [1/4] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 294
7.54.6.2 operator<< [2/4] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 294
7.54.6.3 operator<< [3/4] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 295
7.54.6.4 operator<< [4/4] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 295
7.54.6.5 QEngine . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 295
7.54.7 Member Data Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 295
7.54.7.1 cmat_hash_tbl_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 296
7.54.7.2 count_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 296
7.54.7.3 d_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 296
7.54.7.4 gates_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 296
7.54.7.5 measured_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 296
7.54.7.6 measurement_count_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . 296
7.54.7.7 measurements_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 297
7.54.7.8 name_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 297
7.54.7.9 nc_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 297
7.54.7.10 nq_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 297
7.54.7.11 step_types_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 297
7.55 qpp::QEngine Class Reference . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 298

7.55.1 Detailed Description . . . . . 300
7.55.2 Constructor & Destructor Documentation . . . . . 300
7.55.2.1 QEngine() [1/3] . . . . . 300
7.55.2.2 QEngine() [2/3] . . . . . 300
7.55.2.3 QEngine() [3/3] . . . . . 301
7.55.2.4 ∼QEngine() . . . . . 301
7.55.3 Member Function Documentation . . . . . 301
7.55.3.1 display() . . . . . 301
7.55.3.2 execute() [1/3] . . . . . 301
7.55.3.3 execute() [2/3] . . . . . 302
7.55.3.4 execute() [3/3] . . . . . 302
7.55.3.5 get_circuit() . . . . . 302
7.55.3.6 get_dit() . . . . . 302
7.55.3.7 get_dits() . . . . . 303
7.55.3.8 get_measured() [1/2] . . . . . 303
7.55.3.9 get_measured() [2/2] . . . . . 303
7.55.3.10 get_non_measured() . . . . . 304
7.55.3.11 get_probs() . . . . . 304
7.55.3.12 get_psi() . . . . . 304
7.55.3.13 get_relative_pos_() . . . . . 304
7.55.3.14 operator=() . . . . . 305
7.55.3.15 reset() . . . . . 305
7.55.3.16 set_dit() . . . . . 305
7.55.3.17 set_measured_() . . . . . 306
7.55.3.18 set_psi() . . . . . 306
7.55.3.19 to_JSON() . . . . . 306
7.55.4 Member Data Documentation . . . . . 307
7.55.4.1 dits_ . . . . . 307
7.55.4.2 probs_ . . . . . 307
7.55.4.3 psi_ . . . . . 307

7.55.4.4 qc_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 307
7.55.4.5 subsys_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 308
7.56 qpp::QNoisyEngine< NoiseModel > Class Template Reference . . . . . . . . . . . . . . . . . . 308
7.56.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . 309
7.56.2 Constructor & Destructor Documentation . . . . . . . . . . . . . . . . . . . . . . . . . 309
7.56.2.1 QNoisyEngine() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 310
7.56.3 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . 310
7.56.3.1 execute() [1/4] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 310
7.56.3.2 execute() [2/4] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 310
7.56.3.3 execute() [3/4] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 310
7.56.3.4 execute() [4/4] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 311
7.56.3.5 get_noise_results() . . . . . . . . . . . . . . . . . . . . . . . . . . . 311
7.56.4 Member Data Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . 311
7.56.4.1 noise_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 311
7.56.4.2 noise_results_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 312
7.57 qpp::QubitAmplitudeDampingNoise Class Reference . . . . . . . . . . . . . . . . . . . . . . . . 312
7.57.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . 313
7.57.2 Constructor & Destructor Documentation . . . . . . . . . . . . . . . . . . . . . . . . . 313
7.57.2.1 QubitAmplitudeDampingNoise() . . . . . . . . . . . . . . . . . . . . . . . 313
7.58 qpp::QubitBitFlipNoise Class Reference . . . . . . . . . . . . . . . . . . . . . . . . . . 313
7.58.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . 314
7.58.2 Constructor & Destructor Documentation . . . . . . . . . . . . . . . . . . . . . . . . . 315
7.58.2.1 QubitBitFlipNoise() . . . . . . . . . . . . . . . . . . . . . . . . . . . 315
7.59 qpp::QubitBitPhaseFlipNoise Class Reference . . . . . . . . . . . . . . . . . . . . . . . . 315
7.59.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . 316
7.59.2 Constructor & Destructor Documentation . . . . . . . . . . . . . . . . . . . . . . . . . 316
7.59.2.1 QubitBitPhaseFlipNoise() . . . . . . . . . . . . . . . . . . . . . . . . . 316
7.60 qpp::QubitDepolarizingNoise Class Reference . . . . . . . . . . . . . . . . . . . . . . . . 317
7.60.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . 318
7.60.2 Constructor & Destructor Documentation . . . . . . . . . . . . . . . . . . . . . . . . . 318

7.60.2.1 QubitDepolarizingNoise() . . . . . 318
7.61 qpp::QubitPhaseDampingNoise Class Reference . . . . . 318
7.61.1 Detailed Description . . . . . 319
7.61.2 Constructor & Destructor Documentation . . . . . 319
7.61.2.1 QubitPhaseDampingNoise() . . . . . 319
7.62 qpp::QubitPhaseFlipNoise Class Reference . . . . . 320
7.62.1 Detailed Description . . . . . 321
7.62.2 Constructor & Destructor Documentation . . . . . 321
7.62.2.1 QubitPhaseFlipNoise() . . . . . 321
7.63 qpp::exception::QuditAlreadyMeasured Class Reference . . . . . 321
7.63.1 Detailed Description . . . . . 322
7.63.2 Member Function Documentation . . . . . 322
7.63.2.1 Exception() . . . . . 322
7.63.2.2 type_description() . . . . . 323
7.64 qpp::QuditDepolarizingNoise Class Reference . . . . . 323
7.64.1 Detailed Description . . . . . 324
7.64.2 Constructor & Destructor Documentation . . . . . 324
7.64.2.1 QuditDepolarizingNoise() . . . . . 324
7.64.3 Member Function Documentation . . . . . 325
7.64.3.1 fill_Ks_() . . . . . 325
7.64.3.2 fill_probs_() . . . . . 325
7.65 qpp::RandomDevices Class Reference . . . . . 326
7.65.1 Detailed Description . . . . . 327
7.65.2 Constructor & Destructor Documentation . . . . . 327
7.65.2.1 RandomDevices() . . . . . 327
7.65.2.2 ∼RandomDevices() . . . . . 328
7.65.3 Member Function Documentation . . . . . 328
7.65.3.1 get_prng() . . . . . 328
7.65.3.2 load() . . . . . 328
7.65.3.3 save() . . . . . 328

7.65.4 Friends And Related Function Documentation . . . . . 329
7.65.4.1 internal::Singleton< RandomDevices > . . . . . 329
7.65.5 Member Data Documentation . . . . . 329
7.65.5.1 prng_ . . . . . 329
7.65.5.2 rd_ . . . . . 329
7.66 qpp::internal::Singleton< T > Class Template Reference . . . . . 329
7.66.1 Detailed Description . . . . . 330
7.66.2 Constructor & Destructor Documentation . . . . . 330
7.66.2.1 Singleton() [1/2] . . . . . 331
7.66.2.2 Singleton() [2/2] . . . . . 331
7.66.2.3 ∼Singleton() . . . . . 331
7.66.3 Member Function Documentation . . . . . 331
7.66.3.1 get_instance() . . . . . 331
7.66.3.2 get_thread_local_instance() . . . . . 331
7.66.3.3 operator=() . . . . . 331
7.67 qpp::exception::SizeMismatch Class Reference . . . . . 332
7.67.1 Detailed Description . . . . . 333
7.67.2 Member Function Documentation . . . . . 333
7.67.2.1 Exception() . . . . . 333
7.67.2.2 type_description() . . . . . 333
7.68 qpp::NoiseType::StateDependent Class Reference . . . . . 334
7.68.1 Detailed Description . . . . . 334
7.69 qpp::NoiseType::StateIndependent Class Reference . . . . . 334
7.69.1 Detailed Description . . . . . 334
7.70 qpp::States Class Reference . . . . . 334
7.70.1 Detailed Description . . . . . 336
7.70.2 Constructor & Destructor Documentation . . . . . 337
7.70.2.1 States() . . . . . 337
7.70.2.2 ∼States() . . . . . 337
7.70.3 Member Function Documentation . . . . . 337

7.70.3.1 jn() . . . . . . 337
7.70.3.2 mes() . . . . . . 337
7.70.3.3 minus() . . . . . . 338
7.70.3.4 one() . . . . . . 338
7.70.3.5 plus() . . . . . . 339
7.70.3.6 zero() . . . . . . 339
7.70.4 Friends And Related Function Documentation . . . . . . 339
7.70.4.1 internal::Singleton< const States > . . . . . . 339
7.70.5 Member Data Documentation . . . . . . 339
7.70.5.1 b00 . . . . . . 340
7.70.5.2 b01 . . . . . . 340
7.70.5.3 b10 . . . . . . 340
7.70.5.4 b11 . . . . . . 340
7.70.5.5 GHZ . . . . . . 340
7.70.5.6 pb00 . . . . . . 340
7.70.5.7 pb01 . . . . . . 341
7.70.5.8 pb10 . . . . . . 341
7.70.5.9 pb11 . . . . . . 341
7.70.5.10 pGHZ . . . . . . 341
7.70.5.11 pW . . . . . . 341
7.70.5.12 px0 . . . . . . 341
7.70.5.13 px1 . . . . . . 342
7.70.5.14 py0 . . . . . . 342
7.70.5.15 py1 . . . . . . 342
7.70.5.16 pz0 . . . . . . 342
7.70.5.17 pz1 . . . . . . 342
7.70.5.18 W . . . . . . 342
7.70.5.19 x0 . . . . . . 343
7.70.5.20 x1 . . . . . . 343
7.70.5.21 y0 . . . . . . 343

7.70.5.22 y1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 343
7.70.5.23 z0 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 343
7.70.5.24 z1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 343
7.71 qpp::exception::SubsysMismatchDims Class Reference . . . . . . . . . . . . . . . . . . . . 344
7.71.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 345
7.71.2 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . 345
7.71.2.1 Exception() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 345
7.71.2.2 type_description() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 345
7.72 qpp::Timer< T, CLOCK_T > Class Template Reference . . . . . . . . . . . . . . . . . . . . 346
7.72.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 347
7.72.2 Constructor & Destructor Documentation . . . . . . . . . . . . . . . . . . . . . . . . 347
7.72.2.1 Timer() [1/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 347
7.72.2.2 Timer() [2/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 348
7.72.2.3 Timer() [3/3] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 348
7.72.2.4 ∼Timer() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 348
7.72.3 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . 348
7.72.3.1 display() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 348
7.72.3.2 get_duration() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 349
7.72.3.3 operator=() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 349
7.72.3.4 operator=() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 349
7.72.3.5 tic() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 350
7.72.3.6 tics() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 350
7.72.3.7 toc() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 350
7.72.4 Member Data Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 350
7.72.4.1 end_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 350
7.72.4.2 start_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 351
7.73 qpp::exception::TypeMismatch Class Reference . . . . . . . . . . . . . . . . . . . . . . . 351
7.73.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 352
7.73.2 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . 352
7.73.2.1 Exception() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 352

7.73.2.2 type_description() . . . . . . . . . . . . . . . . . . . . . . . . . . . . 353
7.74 qpp::exception::UndefinedType Class Reference . . . . . . . . . . . . . . . . . . . . . . . . . . 353
7.74.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 354
7.74.2 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . 354
7.74.2.1 Exception() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 354
7.74.2.2 type_description() . . . . . . . . . . . . . . . . . . . . . . . . . . . . 355
7.75 qpp::exception::Unknown Class Reference . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 355
7.75.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 356
7.75.2 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . 356
7.75.2.1 Exception() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 356
7.75.2.2 type_description() . . . . . . . . . . . . . . . . . . . . . . . . . . . . 357
7.76 qpp::QCircuit::iterator::value_type_ Class Reference . . . . . . . . . . . . . . . . . . . . . . . 357
7.76.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 358
7.76.2 Constructor & Destructor Documentation . . . . . . . . . . . . . . . . . . . . . . . . 358
7.76.2.1 value_type_() [1/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . 358
7.76.2.2 value_type_() [2/2] . . . . . . . . . . . . . . . . . . . . . . . . . . . . 359
7.76.3 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . 359
7.76.3.1 display() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 359
7.76.3.2 operator=() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 359
7.76.4 Member Data Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 359
7.76.4.1 gates_ip_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 360
7.76.4.2 ip_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 360
7.76.4.3 measurements_ip_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 360
7.76.4.4 type_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 360
7.76.4.5 value_type_qc_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 360
7.77 qpp::exception::ZeroSize Class Reference . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 361
7.77.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 362
7.77.2 Member Function Documentation . . . . . . . . . . . . . . . . . . . . . . . . . . . . 362
7.77.2.1 Exception() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 362
7.77.2.2 type_description() . . . . . . . . . . . . . . . . . . . . . . . . . . . . 362

**8 File Documentation** 363

8.1 classes/circuits/circuits.h File Reference . . . . . . . . . . 363

8.1.1 Detailed Description . . . . . . . . . . 364

8.2 classes/circuits/engines.h File Reference . . . . . . . . . . 364

8.2.1 Detailed Description . . . . . . . . . . 364

8.3 classes/codes.h File Reference . . . . . . . . . . 365

8.3.1 Detailed Description . . . . . . . . . . 365

8.4 classes/exception.h File Reference . . . . . . . . . . 365

8.4.1 Detailed Description . . . . . . . . . . 367

8.5 classes/gates.h File Reference . . . . . . . . . . 367

8.5.1 Detailed Description . . . . . . . . . . 368

8.6 classes/idisplay.h File Reference . . . . . . . . . . 368

8.6.1 Detailed Description . . . . . . . . . . 369

8.7 classes/init.h File Reference . . . . . . . . . . 369

8.7.1 Detailed Description . . . . . . . . . . 369

8.8 classes/noise.h File Reference . . . . . . . . . . 370

8.8.1 Detailed Description . . . . . . . . . . 370

8.9 classes/random_devices.h File Reference . . . . . . . . . . 371

8.9.1 Detailed Description . . . . . . . . . . 371

8.10 classes/reversible.h File Reference . . . . . . . . . . 371

8.10.1 Detailed Description . . . . . . . . . . 372

8.11 classes/states.h File Reference . . . . . . . . . . 372

8.11.1 Detailed Description . . . . . . . . . . 373

8.12 classes/timer.h File Reference . . . . . . . . . . 373

8.12.1 Detailed Description . . . . . . . . . . 373

8.13 constants.h File Reference . . . . . . . . . . 374

8.13.1 Detailed Description . . . . . . . . . . 375

8.14 entanglement.h File Reference . . . . . . . . . . 375

8.14.1 Detailed Description . . . . . . . . . . 376

8.15 entropies.h File Reference . . . . . . . . . . 376

8.15.1 Detailed Description . . . . . . . . . . 377
8.16 experimental/experimental.h File Reference . . . . . . . . . . 378
8.16.1 Detailed Description . . . . . . . . . . 378
8.17 functions.h File Reference . . . . . . . . . . 378
8.17.1 Detailed Description . . . . . . . . . . 383
8.18 input_output.h File Reference . . . . . . . . . . 383
8.18.1 Detailed Description . . . . . . . . . . 384
8.19 instruments.h File Reference . . . . . . . . . . 384
8.19.1 Detailed Description . . . . . . . . . . 385
8.20 internal/classes/iomanip.h File Reference . . . . . . . . . . 385
8.20.1 Detailed Description . . . . . . . . . . 386
8.21 internal/classes/singleton.h File Reference . . . . . . . . . . 386
8.21.1 Detailed Description . . . . . . . . . . 387
8.22 internal/util.h File Reference . . . . . . . . . . 387
8.22.1 Detailed Description . . . . . . . . . . 388
8.23 MATLAB/matlab.h File Reference . . . . . . . . . . 389
8.23.1 Detailed Description . . . . . . . . . . 389
8.24 number_theory.h File Reference . . . . . . . . . . 389
8.24.1 Detailed Description . . . . . . . . . . 391
8.25 operations.h File Reference . . . . . . . . . . 391
8.25.1 Detailed Description . . . . . . . . . . 393
8.26 qpp.h File Reference . . . . . . . . . . 393
8.26.1 Detailed Description . . . . . . . . . . 395
8.26.2 Macro Definition Documentation . . . . . . . . . . 395
8.26.2.1 QPP_UNUSED_ . . . . . . . . . . 395
8.27 random.h File Reference . . . . . . . . . . 395
8.27.1 Detailed Description . . . . . . . . . . 397
8.28 statistics.h File Reference . . . . . . . . . . 397
8.28.1 Detailed Description . . . . . . . . . . 398
8.29 traits.h File Reference . . . . . . . . . . 398
8.29.1 Detailed Description . . . . . . . . . . 399
8.30 types.h File Reference . . . . . . . . . . 399
8.30.1 Detailed Description . . . . . . . . . . 400

**Index** **401**

# Chapter 1

# Quantum++

### Version 1.2 - 10 February 2019

**Build status:**

**Chat (questions/issues)**

## About

Quantum++ is a modern C++11 general purpose quantum computing library, composed solely of template header files. Quantum++ is written in standard C++11 and has very low external dependencies, using only the Eigen 3 linear algebra header-only template library and, if available, the OpenMP multi-processing library.

Quantum++ is not restricted to qubit systems or specific quantum information processing tasks, being capable of simulating arbitrary quantum processes. The main design factors taken in consideration were the ease of use, high portability, and high performance. The library's simulation capabilities are only restricted by the amount of available physical memory. On a typical machine (Intel i5 8Gb RAM) Quantum++ can successfully simulate the evolution of 25 qubits in a pure state or of 12 qubits in a mixed state reasonably fast.

To report any bugs or ask for additional features/enhancements, please submit an issue with an appropriate label.

If you are interesting in contributing to this project, feel free to contact me. Alternatively, create a custom branch, add your contribution, then finally create a pull request. If I accept the pull request, I will merge your custom branch with the latest development branch. The latter will eventually be merged into a future release version. To contribute, you need to have a solid knowledge of C++ (preferably C++11), including templates and the standard library, a basic knowledge of quantum computing and linear algebra, and working experience with Eigen 3.

For additional Eigen 3 documentation see http://eigen.tuxfamily.org/dox/. For a simple Eigen 3 quick ASCII reference see http://eigen.tuxfamily.org/dox/AsciiQuickReference.txt.



## License

Quantum++ is distributed under the MIT license. Please see the LICENSE file for more details.

## Installation instructions and further documentation

Please see the installation guide https://github.com/vsoftco/qpp/blob/master/INSTALL.md "`INSTALL.md`" and the comprehensive Wiki for further documentation and detailed examples.

The official API documentation is available in PDF and HTML formats in the doc folder.

# Chapter 2

# Namespace Index

## 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

qpp
Quantum++ main namespace . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
qpp::exception
Quantum++ exception hierarchy namespace . . . . . . . . . . . . . . . . . . . . . . . . . . 116
qpp::experimental
Experimental/test functions/classes, do not use or modify . . . . . . . . . . . . . . . . . . . . 118
qpp::internal
Internal utility functions, do not use them directly or modify them . . . . . . . . . . . . . . . . 118
qpp::literals . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 125

# Chapter 3

# Hierarchical Index

## 3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:


qpp::internal::Display_Impl_ . . . . . 155
qpp::internal::IOManipEigen . . . . . 199
qpp::internal::EqualEigen . . . . . 169
std::exception
qpp::exception::Exception . . . . . 170
qpp::exception::CustomException . . . . . 140
qpp::exception::DimsInvalid . . . . . 143
qpp::exception::DimsMismatchCvector . . . . . 145
qpp::exception::DimsMismatchMatrix . . . . . 147
qpp::exception::DimsMismatchRvector . . . . . 149
qpp::exception::DimsMismatchVector . . . . . 151
qpp::exception::DimsNotEqual . . . . . 153
qpp::exception::Duplicates . . . . . 156
qpp::exception::InvalidIterator . . . . . 197
qpp::exception::MatrixMismatchSubsys . . . . . 219
qpp::exception::MatrixNotCvector . . . . . 221
qpp::exception::MatrixNotRvector . . . . . 223
qpp::exception::MatrixNotSquare . . . . . 225
qpp::exception::MatrixNotSquareNorCvector . . . . . 227
qpp::exception::MatrixNotSquareNorRvector . . . . . 229
qpp::exception::MatrixNotSquareNorVector . . . . . 231
qpp::exception::MatrixNotVector . . . . . 233
qpp::exception::NoCodeword . . . . . 238
qpp::exception::NotBipartite . . . . . 248
qpp::exception::NotImplemented . . . . . 250
qpp::exception::NotQubitCvector . . . . . 252
qpp::exception::NotQubitMatrix . . . . . 254
qpp::exception::NotQubitRvector . . . . . 256
qpp::exception::NotQubitSubsys . . . . . 258
qpp::exception::NotQubitVector . . . . . 260
qpp::exception::OutOfRange . . . . . 262
qpp::exception::PermInvalid . . . . . 264
qpp::exception::PermMismatchDims . . . . . 266
qpp::exception::QuditAlreadyMeasured . . . . . 321
qpp::exception::SizeMismatch . . . . . 332

qpp::exception::SubsysMismatchDims . . . . . 344
qpp::exception::TypeMismatch . . . . . 351
qpp::exception::UndefinedType . . . . . 353
qpp::exception::Unknown . . . . . 355
qpp::exception::ZeroSize . . . . . 361
false_type
qpp::is_complex< T > . . . . . 208
qpp::is_iterable< T, typename > . . . . . 210
qpp::QCircuit::GateStep . . . . . 186
qpp::internal::HashEigen . . . . . 188
qpp::IDisplay . . . . . 189
qpp::Dynamic_bitset . . . . . 158
qpp::Bit_circuit . . . . . 129
qpp::internal::IOManipEigen . . . . . 199
qpp::internal::IOManipPointer< PointerType > . . . . . 201
qpp::internal::IOManipRange< InputIterator > . . . . . 205
qpp::QCircuit . . . . . 268
qpp::QCircuit::iterator::value_type_ . . . . . 357
qpp::QEngine . . . . . 298
qpp::QNoisyEngine< NoiseModel > . . . . . 308
qpp::Timer< T, CLOCK_T > . . . . . 346
qpp::IJSON . . . . . 192
qpp::QCircuit . . . . . 268
qpp::QEngine . . . . . 298
is_base_of
qpp::is_matrix_expression< Derived > . . . . . 212
qpp::QCircuit::iterator . . . . . 213
qpp::make_void< Ts > . . . . . 218
qpp::QCircuit::MeasureStep . . . . . 235
qpp::NoiseBase< T > . . . . . 240
qpp::NoiseBase< NoiseType::StateDependent > . . . . . 240
qpp::QubitAmplitudeDampingNoise . . . . . 312
qpp::QubitPhaseDampingNoise . . . . . 318
qpp::NoiseBase< NoiseType::StateIndependent > . . . . . 240
qpp::QubitBitFlipNoise . . . . . 313
qpp::QubitBitPhaseFlipNoise . . . . . 315
qpp::QubitDepolarizingNoise . . . . . 317
qpp::QubitPhaseFlipNoise . . . . . 320
qpp::QuditDepolarizingNoise . . . . . 323
qpp::NoiseType . . . . . 247
qpp::internal::Singleton< T > . . . . . 329
qpp::internal::Singleton< const Codes > . . . . . 329
qpp::Codes . . . . . 137
qpp::internal::Singleton< const Gates > . . . . . 329
qpp::Gates . . . . . 173
qpp::internal::Singleton< const Init > . . . . . 329
qpp::Init . . . . . 195
qpp::internal::Singleton< const States > . . . . . 329
qpp::States . . . . . 334
qpp::internal::Singleton< RandomDevices > . . . . . 329
qpp::RandomDevices . . . . . 326
qpp::NoiseType::StateDependent . . . . . 334
qpp::NoiseType::StateIndependent . . . . . 334
true_type
qpp::is_complex< std::complex< T > > . . . . . 209
qpp::is_iterable< T, to_void< decltype(std::declval< T >().begin()), decltype(std::declval< T >().end()), decltype(*(std::declval< T >().begin()))> > . . . . . 211

# Chapter 4

# Class Index

## 4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:


qpp::Bit_circuit
Classical reversible circuit simulator . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 129
qpp::Codes
Const Singleton class that defines quantum error correcting codes . . . . . . . . . . . . . . . 137
qpp::exception::CustomException
Custom exception . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 140
qpp::exception::DimsInvalid
Invalid dimension(s) exception . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 143
qpp::exception::DimsMismatchCvector
Dimension(s) mismatch column vector size exception . . . . . . . . . . . . . . . . . . . . . 145
qpp::exception::DimsMismatchMatrix
Dimension(s) mismatch matrix size exception . . . . . . . . . . . . . . . . . . . . . . . . . 147
qpp::exception::DimsMismatchRvector
Dimension(s) mismatch row vector size exception . . . . . . . . . . . . . . . . . . . . . . . 149
qpp::exception::DimsMismatchVector
Dimension(s) mismatch vector size exception . . . . . . . . . . . . . . . . . . . . . . . . . 151
qpp::exception::DimsNotEqual
Dimensions not equal exception . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 153
qpp::internal::Display_Impl_ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 155
qpp::exception::Duplicates
System (e.g. std::vector) has duplicates exception . . . . . . . . . . . . . . . . . . . . . . 156
qpp::Dynamic_bitset
Dynamic bitset class, allows the specification of the number of bits at runtime . . . . . . . . . 158
qpp::internal::EqualEigen
Functor for comparing Eigen expressions for equality . . . . . . . . . . . . . . . . . . . . . 169
qpp::exception::Exception
Base class for generating Quantum++ custom exceptions . . . . . . . . . . . . . . . . . . . 170
qpp::Gates
Const Singleton class that implements most commonly used gates . . . . . . . . . . . . . . . 173
qpp::QCircuit::GateStep
One step consisting only of gates/operators in the circuit . . . . . . . . . . . . . . . . . . . 186
qpp::internal::HashEigen
Functor for hashing Eigen expressions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 188
qpp::IDisplay
Abstract class (interface) that mandates the definition of virtual std::ostream& display(std::ostream& os) const
189

qpp::IJSON
Abstract class (interface) that mandates the definition of very basic JSON serialization support 192
qpp::Init
Const Singleton class that performs additional initializations/cleanups . . . . . . . . . . . . . 195
qpp::exception::InvalidIterator
Invalid iterator . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 197
qpp::internal::IOManipEigen . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 199
qpp::internal::IOManipPointer< PointerType > . . . . . . . . . . . . . . . . . . . . . . . . . . 201
qpp::internal::IOManipRange< InputIterator > . . . . . . . . . . . . . . . . . . . . . . . . . . 205
qpp::is_complex< T >
Checks whether the type is a complex type . . . . . . . . . . . . . . . . . . . . . . . . . . 208
qpp::is_complex< std::complex< T > >
Checks whether the type is a complex number type, specialization for complex types . . . . . 209
qpp::is_iterable< T, typename >
Checks whether *T* is compatible with an STL-like iterable container . . . . . . . . . . . . . . . 210
qpp::is_iterable< T, to_void< decltype(std::declval< T >().begin()), decltype(std::declval< T >().end()), decltype(∗(std::declval<
Checks whether *T* is compatible with an STL-like iterable container, specialization for STL-like iterable containers . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 211
qpp::is_matrix_expression< Derived >
Checks whether the type is an Eigen matrix expression . . . . . . . . . . . . . . . . . . . . . 212
qpp::QCircuit::iterator
Quantum circuit bound-checking (safe) iterator . . . . . . . . . . . . . . . . . . . . . . . . . 213
qpp::make_void< Ts >
Helper for qpp::to_void<> alias template . . . . . . . . . . . . . . . . . . . . . . . . . . . . 218
qpp::exception::MatrixMismatchSubsys
Matrix mismatch subsystems exception . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 219
qpp::exception::MatrixNotCvector
Matrix is not a column vector exception . . . . . . . . . . . . . . . . . . . . . . . . . . . . 221
qpp::exception::MatrixNotRvector
Matrix is not a row vector exception . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 223
qpp::exception::MatrixNotSquare
Matrix is not square exception . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 225
qpp::exception::MatrixNotSquareNorCvector
Matrix is not square nor column vector exception . . . . . . . . . . . . . . . . . . . . . . . . 227
qpp::exception::MatrixNotSquareNorRvector
Matrix is not square nor row vector exception . . . . . . . . . . . . . . . . . . . . . . . . . 229
qpp::exception::MatrixNotSquareNorVector
Matrix is not square nor vector exception . . . . . . . . . . . . . . . . . . . . . . . . . . . . 231
qpp::exception::MatrixNotVector
Matrix is not a vector exception . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 233
qpp::QCircuit::MeasureStep
One step consisting only of measurements in the circuit . . . . . . . . . . . . . . . . . . . . 235
qpp::exception::NoCodeword
Codeword does not exist exception . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 238
qpp::NoiseBase< T >
Base class for all noise models, derive your particular noise model . . . . . . . . . . . . . . . 240
qpp::NoiseType
Contains template tags used to specify the noise type . . . . . . . . . . . . . . . . . . . . . 247
qpp::exception::NotBipartite
Not bi-partite exception . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 248
qpp::exception::NotImplemented
Code not yet implemented . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 250
qpp::exception::NotQubitCvector
Column vector is not 2 x 1 exception . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 252
qpp::exception::NotQubitMatrix
Matrix is not 2 x 2 exception . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 254
qpp::exception::NotQubitRvector
Row vector is not 1 x 2 exception . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 256

qpp::exception::NotQubitSubsys
Subsystems are not qubits exception . . . . . 258
qpp::exception::NotQubitVector
Vector is not 2 x 1 nor 1 x 2 exception . . . . . 260
qpp::exception::OutOfRange
Argument out of range exception . . . . . 262
qpp::exception::PermInvalid
Invalid permutation exception . . . . . 264
qpp::exception::PermMismatchDims
Permutation mismatch dimensions exception . . . . . 266
qpp::QCircuit
Quantum circuit class . . . . . 268
qpp::QEngine
Quantum circuit engine, executes qpp::QCircuit . . . . . 298
qpp::QNoisyEngine< NoiseModel >
Noisy quantum circuit engine, executes qpp::QCircuit . . . . . 308
qpp::QubitAmplitudeDampingNoise
Qubit amplitude damping noise, as described in Nielsen and Chuang . . . . . 312
qpp::QubitBitFlipNoise
Qubit bit flip noise . . . . . 313
qpp::QubitBitPhaseFlipNoise
Qubit bit-phase flip (dephasing) noise . . . . . 315
qpp::QubitDepolarizingNoise
Qubit depolarizing noise . . . . . 317
qpp::QubitPhaseDampingNoise
Qubit phase damping noise, as described in Nielsen and Chuang . . . . . 318
qpp::QubitPhaseFlipNoise
Qubit phase flip (dephasing) noise . . . . . 320
qpp::exception::QuditAlreadyMeasured
Qudit was already measured exception . . . . . 321
qpp::QuditDepolarizingNoise
Qudit depolarizing noise . . . . . 323
qpp::RandomDevices
Singleton class that manages the source of randomness in the library . . . . . 326
qpp::internal::Singleton< T >
Singleton policy class, used internally to implement the singleton pattern via CRTP (Curiously recurring template pattern) . . . . . 329
qpp::exception::SizeMismatch
Size mismatch exception . . . . . 332
qpp::NoiseType::StateDependent
Template tag, used whenever the noise is state-dependent . . . . . 334
qpp::NoiseType::StateIndependent
Template tag, used whenever the noise is state-independent . . . . . 334
qpp::States
Const Singleton class that implements most commonly used states . . . . . 334
qpp::exception::SubsysMismatchDims
Subsystems mismatch dimensions exception . . . . . 344
qpp::Timer< T, CLOCK_T >
Chronometer . . . . . 346
qpp::exception::TypeMismatch
Type mismatch exception . . . . . 351
qpp::exception::UndefinedType
Not defined for this type exception . . . . . 353
qpp::exception::Unknown
Unknown exception . . . . . 355
qpp::QCircuit::iterator::value_type_
Value type class for qpp::QCircuit::iterator . . . . . 357

qpp::exception::ZeroSize
Object has zero size exception . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 361

# Chapter 5

# File Index

## 5.1 File List

Here is a list of all files with brief descriptions:


constants.h
Constants . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 374
entanglement.h
Entanglement functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 375
entropies.h
Entropy functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 376
functions.h
Generic quantum computing functions . . . . . . . . . . . . . . . . . . . . . . . . . 378
input_output.h
Input/output functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 383
instruments.h
Measurement functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 384
number_theory.h
Number theory functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 389
operations.h
Quantum operation functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 391
qpp.h
Quantum++ main header file, includes all other necessary headers . . . . . . . . . . 393
random.h
Randomness-related functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 395
statistics.h
Statistics functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 397
traits.h
Type traits . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 398
types.h
Type aliases . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 399
classes/codes.h
Quantum error correcting codes . . . . . . . . . . . . . . . . . . . . . . . . . . . . 365
classes/exception.h
Exceptions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 365
classes/gates.h
Quantum gates . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 367
classes/idisplay.h
Display interface via the non-virtual interface (NVI) and very basic JSON serialization support interface . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 368

classes/init.h
Initialization . . . . . 369
classes/noise.h
Noise models . . . . . 370
classes/random_devices.h
Random devices . . . . . 371
classes/reversible.h
Support for classical reversible circuits . . . . . 371
classes/states.h
Quantum states . . . . . 372
classes/timer.h
Timing . . . . . 373
classes/circuits/circuits.h
Qudit quantum circuits . . . . . 363
classes/circuits/engines.h
Qudit quantum engines . . . . . 364
experimental/experimental.h
Experimental/test functions/classes . . . . . 378
internal/util.h
Internal utility functions . . . . . 387
internal/classes/iomanip.h
Input/output manipulators . . . . . 385
internal/classes/singleton.h
Singleton pattern via CRTP . . . . . 386
MATLAB/matlab.h
Input/output interfacing with MATLAB . . . . . 389

# Chapter 6

# Namespace Documentation

## 6.1 qpp Namespace Reference

Quantum++ main namespace.

### Namespaces

- exception
  *Quantum++ exception hierarchy namespace.*
- experimental
  *Experimental/test functions/classes, do not use or modify.*
- internal
  *Internal utility functions, do not use them directly or modify them.*
- literals

### Classes

- class Bit_circuit
  *Classical reversible circuit simulator.*
- class Codes
  *const Singleton class that defines quantum error correcting codes*
- class Dynamic_bitset
  *Dynamic bitset class, allows the specification of the number of bits at runtime.*
- class Gates
  *const Singleton class that implements most commonly used gates*
- class IDisplay
  *Abstract class (interface) that mandates the definition of virtual std::ostream& display(std::ostream& os) const.*
- class IJSON
  *Abstract class (interface) that mandates the definition of very basic JSON serialization support.*
- class Init
  *const Singleton class that performs additional initializations/cleanups*
- struct is_complex
  *Checks whether the type is a complex type.*
- struct is_complex< std::complex< T > >

*Checks whether the type is a complex number type, specialization for complex types.*
- struct is_iterable

  *Checks whether T is compatible with an STL-like iterable container.*
- struct is_iterable< T, to_void< decltype(std::declval< T >().begin()), decltype(std::declval< T >().end()), decltype(∗(std::declv

  *Checks whether T is compatible with an STL-like iterable container, specialization for STL-like iterable containers.*
- struct is_matrix_expression

  *Checks whether the type is an Eigen matrix expression.*
- struct make_void

  *Helper for qpp::to_void<> alias template.*
- class NoiseBase

  *Base class for all noise models, derive your particular noise model.*
- class NoiseType

  *Contains template tags used to specify the noise type.*
- class QCircuit

  *Quantum circuit class.*
- class QEngine

  *Quantum circuit engine, executes qpp::QCircuit.*
- class QNoisyEngine

  *Noisy quantum circuit engine, executes qpp::QCircuit.*
- class QubitAmplitudeDampingNoise

  *Qubit amplitude damping noise, as described in Nielsen and Chuang.*
- class QubitBitFlipNoise

  *Qubit bit flip noise.*
- class QubitBitPhaseFlipNoise

  *Qubit bit-phase flip (dephasing) noise.*
- class QubitDepolarizingNoise

  *Qubit depolarizing noise.*
- class QubitPhaseDampingNoise

  *Qubit phase damping noise, as described in Nielsen and Chuang.*
- class QubitPhaseFlipNoise

  *Qubit phase flip (dephasing) noise.*
- class QuditDepolarizingNoise

  *Qudit depolarizing noise.*
- class RandomDevices

  *Singleton class that manages the source of randomness in the library.*
- class States

  *const Singleton class that implements most commonly used states*
- class Timer

  *Chronometer.*

## Typedefs

- template<typename... Ts>
  using to_void = typename make_void< Ts... >::type

  *Alias template that implements the proposal for void_t.*
- using idx = std::size_t

  *Non-negative integer index, make sure you use an unsigned type.*
- using bigint = long long int

  *Big integer.*
- using cplx = std::complex< double >

*Complex number in double precision.*

- using ket = Eigen::VectorXcd

  *Complex (double precision) dynamic Eigen column vector.*
- using bra = Eigen::RowVectorXcd

  *Complex (double precision) dynamic Eigen row vector.*
- using cmat = Eigen::MatrixXcd

  *Complex (double precision) dynamic Eigen matrix.*
- using dmat = Eigen::MatrixXd

  *Real (double precision) dynamic Eigen matrix.*
- template< typename Scalar >
  using dyn_mat = Eigen::Matrix< Scalar, Eigen::Dynamic, Eigen::Dynamic >

  *Dynamic Eigen matrix over the field specified by Scalar.*
- template< typename Scalar >
  using dyn_col_vect = Eigen::Matrix< Scalar, Eigen::Dynamic, 1 >

  *Dynamic Eigen column vector over the field specified by Scalar.*
- template< typename Scalar >
  using dyn_row_vect = Eigen::Matrix< Scalar, 1, Eigen::Dynamic >

  *Dynamic Eigen row vector over the field specified by Scalar.*

## Functions

- constexpr cplx operator"" _i (long double x) noexcept

  *User-defined literal for complex $i = \sqrt{-1}$ (real overload)*
- cplx omega (idx D)

  *D-th root of unity.*
- template< typename Derived >
  dyn_col_vect< double > schmidtcoeffs (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)

  *Schmidt coefficients of the bi-partite pure state A.*
- template< typename Derived >
  dyn_col_vect< double > schmidtcoeffs (const Eigen::MatrixBase< Derived > &A, idx d=2)

  *Schmidt coefficients of the bi-partite pure state A.*
- template< typename Derived >
  cmat schmidtA (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)

  *Schmidt basis on Alice side.*
- template< typename Derived >
  cmat schmidtA (const Eigen::MatrixBase< Derived > &A, idx d=2)

  *Schmidt basis on Alice side.*
- template< typename Derived >
  cmat schmidtB (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)

  *Schmidt basis on Bob side.*
- template< typename Derived >
  cmat schmidtB (const Eigen::MatrixBase< Derived > &A, idx d=2)

  *Schmidt basis on Bob side.*
- template< typename Derived >
  std::vector< double > schmidtprobs (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)

  *Schmidt probabilities of the bi-partite pure state A.*
- template< typename Derived >
  std::vector< double > schmidtprobs (const Eigen::MatrixBase< Derived > &A, idx d=2)

  *Schmidt probabilities of the bi-partite pure state A.*

- template< typename Derived >
  double entanglement (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)

    *Entanglement of the bi-partite pure state A.*
- template< typename Derived >
  double entanglement (const Eigen::MatrixBase< Derived > &A, idx d=2)

    *Entanglement of the bi-partite pure state A.*
- template< typename Derived >
  double gconcurrence (const Eigen::MatrixBase< Derived > &A)

    *G-concurrence of the bi-partite pure state A.*
- template< typename Derived >
  double negativity (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)

    *Negativity of the bi-partite mixed state A.*
- template< typename Derived >
  double negativity (const Eigen::MatrixBase< Derived > &A, idx d=2)

    *Negativity of the bi-partite mixed state A.*
- template< typename Derived >
  double lognegativity (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)

    *Logarithmic negativity of the bi-partite mixed state A.*
- template< typename Derived >
  double lognegativity (const Eigen::MatrixBase< Derived > &A, idx d=2)

    *Logarithmic negativity of the bi-partite mixed state A.*
- template< typename Derived >
  double concurrence (const Eigen::MatrixBase< Derived > &A)

    *Wootters concurrence of the bi-partite qubit mixed state A.*
- template< typename Derived >
  double entropy (const Eigen::MatrixBase< Derived > &A)

    *von-Neumann entropy of the density matrix A*
- double entropy (const std::vector< double > &prob)

    *Shannon entropy of the probability distribution prob.*
- template< typename Derived >
  double renyi (const Eigen::MatrixBase< Derived > &A, double alpha)

    *Renyi- $\alpha$ entropy of the density matrix A, for $\alpha \geq 0$.*
- double renyi (const std::vector< double > &prob, double alpha)

    *Renyi- $\alpha$ entropy of the probability distribution prob, for $\alpha \geq 0$.*
- template< typename Derived >
  double tsallis (const Eigen::MatrixBase< Derived > &A, double q)

    *Tsallis- $q$ entropy of the density matrix A, for $q \geq 0$.*
- double tsallis (const std::vector< double > &prob, double q)

    *Tsallis- $q$ entropy of the probability distribution prob, for $q \geq 0$.*
- template< typename Derived >
  double qmutualinfo (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &subsysA, const std::vector< idx > &subsysB, const std::vector< idx > &dims)

    *Quantum mutual information between 2 subsystems of a composite system.*
- template< typename Derived >
  double qmutualinfo (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &subsysA, const std::vector< idx > &subsysB, idx d=2)

    *Quantum mutual information between 2 subsystems of a composite system.*
- template< typename Derived >
  dyn_mat< typename Derived::Scalar > transpose (const Eigen::MatrixBase< Derived > &A)

    *Transpose.*
- template< typename Derived >
  dyn_mat< typename Derived::Scalar > conjugate (const Eigen::MatrixBase< Derived > &A)

    *Complex conjugate.*

- template< typename Derived >
  dyn_mat< typename Derived::Scalar > adjoint (const Eigen::MatrixBase< Derived > &A)

  *Adjoint.*
- template< typename Derived >
  dyn_mat< typename Derived::Scalar > inverse (const Eigen::MatrixBase< Derived > &A)

  *Inverse.*
- template< typename Derived >
  Derived::Scalar trace (const Eigen::MatrixBase< Derived > &A)

  *Trace.*
- template< typename Derived >
  Derived::Scalar det (const Eigen::MatrixBase< Derived > &A)

  *Determinant.*
- template< typename Derived >
  Derived::Scalar logdet (const Eigen::MatrixBase< Derived > &A)

  *Logarithm of the determinant.*
- template< typename Derived >
  Derived::Scalar sum (const Eigen::MatrixBase< Derived > &A)

  *Element-wise sum of A.*
- template< typename Derived >
  Derived::Scalar prod (const Eigen::MatrixBase< Derived > &A)

  *Element-wise product of A.*
- template< typename Derived >
  double norm (const Eigen::MatrixBase< Derived > &A)

  *Frobenius norm.*
- template< typename Derived >
  dyn_mat< typename Derived::Scalar > normalize (const Eigen::MatrixBase< Derived > &A)

  *Normalizes state vector (column or row vector) or density matrix.*
- template< typename Derived >
  std::pair< dyn_col_vect< cplx >, cmat > eig (const Eigen::MatrixBase< Derived > &A)

  *Full eigen decomposition.*
- template< typename Derived >
  dyn_col_vect< cplx > evals (const Eigen::MatrixBase< Derived > &A)

  *Eigenvalues.*
- template< typename Derived >
  cmat evects (const Eigen::MatrixBase< Derived > &A)

  *Eigenvectors.*
- template< typename Derived >
  std::pair< dyn_col_vect< double >, cmat > heig (const Eigen::MatrixBase< Derived > &A)

  *Full eigen decomposition of Hermitian expression.*
- template< typename Derived >
  dyn_col_vect< double > hevals (const Eigen::MatrixBase< Derived > &A)

  *Hermitian eigenvalues.*
- template< typename Derived >
  cmat hevects (const Eigen::MatrixBase< Derived > &A)

  *Eigenvectors of Hermitian matrix.*
- template< typename Derived >
  std::tuple< cmat, dyn_col_vect< double >, cmat > svd (const Eigen::MatrixBase< Derived > &A)

  *Full singular value decomposition.*
- template< typename Derived >
  dyn_col_vect< double > svals (const Eigen::MatrixBase< Derived > &A)

  *Singular values.*
- template< typename Derived >
  cmat svdU (const Eigen::MatrixBase< Derived > &A)

*Left singular vectors.*

- template< typename Derived >
cmat svdV (const Eigen::MatrixBase< Derived > &A)

  *Right singular vectors.*

- template< typename Derived >
cmat funm (const Eigen::MatrixBase< Derived > &A, cplx(*f)(const cplx &))

  *Functional calculus f(A)*

- template< typename Derived >
cmat sqrtm (const Eigen::MatrixBase< Derived > &A)

  *Matrix square root.*

- template< typename Derived >
cmat absm (const Eigen::MatrixBase< Derived > &A)

  *Matrix absolute value.*

- template< typename Derived >
cmat expm (const Eigen::MatrixBase< Derived > &A)

  *Matrix exponential.*

- template< typename Derived >
cmat logm (const Eigen::MatrixBase< Derived > &A)

  *Matrix logarithm.*

- template< typename Derived >
cmat sinm (const Eigen::MatrixBase< Derived > &A)

  *Matrix sin.*

- template< typename Derived >
cmat cosm (const Eigen::MatrixBase< Derived > &A)

  *Matrix cos.*

- template< typename Derived >
cmat spectralpowm (const Eigen::MatrixBase< Derived > &A, const cplx z)

  *Matrix power.*

- template< typename Derived >
dyn_mat< typename Derived::Scalar > powm (const Eigen::MatrixBase< Derived > &A, idx n)

  *Fast matrix power based on the SQUARE-AND-MULTIPLY algorithm.*

- template< typename Derived >
double schatten (const Eigen::MatrixBase< Derived > &A, double p)

  *Schatten matrix norm.*

- template< typename OutputScalar , typename Derived >
dyn_mat< OutputScalar > cwise (const Eigen::MatrixBase< Derived > &A, OutputScalar(*f)(const typename Derived::Scalar &))

  *Functor.*

- template< typename T >
dyn_mat< typename T::Scalar > kron (const T &head)

  *Kronecker product.*

- template< typename T , typename... Args>
dyn_mat< typename T::Scalar > kron (const T &head, const Args &... tail)

  *Kronecker product.*

- template< typename Derived >
dyn_mat< typename Derived::Scalar > kron (const std::vector< Derived > &As)

  *Kronecker product.*

- template< typename Derived >
dyn_mat< typename Derived::Scalar > kron (const std::initializer_list< Derived > &As)

  *Kronecker product.*

- template< typename Derived >
dyn_mat< typename Derived::Scalar > kronpow (const Eigen::MatrixBase< Derived > &A, idx n)

  *Kronecker power.*

- template< typename T >
dyn_mat< typename T::Scalar > dirsum (const T &head)

    *Direct sum.*

- template< typename T , typename... Args>
dyn_mat< typename T::Scalar > dirsum (const T &head, const Args &... tail)

    *Direct sum.*

- template< typename Derived >
dyn_mat< typename Derived::Scalar > dirsum (const std::vector< Derived > &As)

    *Direct sum.*

- template< typename Derived >
dyn_mat< typename Derived::Scalar > dirsum (const std::initializer_list< Derived > &As)

    *Direct sum.*

- template< typename Derived >
dyn_mat< typename Derived::Scalar > dirsumpow (const Eigen::MatrixBase< Derived > &A, idx n)

    *Direct sum power.*

- template< typename Derived >
dyn_mat< typename Derived::Scalar > reshape (const Eigen::MatrixBase< Derived > &A, idx rows, idx cols)

    *Reshape.*

- template< typename Derived1 , typename Derived2 >
dyn_mat< typename Derived1::Scalar > comm (const Eigen::MatrixBase< Derived1 > &A, const Eigen::MatrixBase< Derived2 > &B)

    *Commutator.*

- template< typename Derived1 , typename Derived2 >
dyn_mat< typename Derived1::Scalar > anticomm (const Eigen::MatrixBase< Derived1 > &A, const Eigen::MatrixBase< Derived2 > &B)

    *Anti-commutator.*

- template< typename Derived >
dyn_mat< typename Derived::Scalar > prj (const Eigen::MatrixBase< Derived > &A)

    *Projector.*

- template< typename Derived >
dyn_mat< typename Derived::Scalar > grams (const std::vector< Derived > &As)

    *Gram-Schmidt orthogonalization.*

- template< typename Derived >
dyn_mat< typename Derived::Scalar > grams (const std::initializer_list< Derived > &As)

    *Gram-Schmidt orthogonalization.*

- template< typename Derived >
dyn_mat< typename Derived::Scalar > grams (const Eigen::MatrixBase< Derived > &A)

    *Gram-Schmidt orthogonalization.*

- std::vector< idx > n2multiidx (idx n, const std::vector< idx > &dims)

    *Non-negative integer index to multi-index.*

- idx multiidx2n (const std::vector< idx > &midx, const std::vector< idx > &dims)

    *Multi-index to non-negative integer index.*

- ket mket (const std::vector< idx > &mask, const std::vector< idx > &dims)

    *Multi-partite qudit ket.*

- ket mket (const std::vector< idx > &mask, idx d=2)

    *Multi-partite qudit ket.*

- cmat mprj (const std::vector< idx > &mask, const std::vector< idx > &dims)

    *Projector onto multi-partite qudit ket.*

- cmat mprj (const std::vector< idx > &mask, idx d=2)

    *Projector onto multi-partite qudit ket.*

- template< typename InputIterator >
std::vector< double > abssq (InputIterator first, InputIterator last)

*Computes the absolute values squared of an STL-like range of complex numbers.*

- template< typename Container >
  std::vector< double > abssq (const Container &c, typename std::enable_if< is_iterable< Container >::value >::type *=nullptr)

  *Computes the absolute values squared of an STL-like container.*
- template< typename Derived >
  std::vector< double > abssq (const Eigen::MatrixBase< Derived > &A)

  *Computes the absolute values squared of an Eigen expression.*
- template< typename InputIterator >
  std::iterator_traits< InputIterator >::value_type sum (InputIterator first, InputIterator last)

  *Element-wise sum of an STL-like range.*
- template< typename Container >
  Container::value_type sum (const Container &c, typename std::enable_if< is_iterable< Container >::value >::type *=nullptr)

  *Element-wise sum of the elements of an STL-like container.*
- template< typename InputIterator >
  std::iterator_traits< InputIterator >::value_type prod (InputIterator first, InputIterator last)

  *Element-wise product of an STL-like range.*
- template< typename Container >
  Container::value_type prod (const Container &c, typename std::enable_if< is_iterable< Container >::value >::type *=nullptr)

  *Element-wise product of the elements of an STL-like container.*
- template< typename Derived >
  dyn_col_vect< typename Derived::Scalar > rho2pure (const Eigen::MatrixBase< Derived > &A)

  *Finds the pure state representation of a matrix proportional to a projector onto a pure state.*
- std::vector< idx > complement (std::vector< idx > subsys, idx n)

  *Constructs the complement of a subsystem vector.*
- template< typename Derived >
  std::vector< double > rho2bloch (const Eigen::MatrixBase< Derived > &A)

  *Computes the 3-dimensional real Bloch vector corresponding to the qubit density matrix A.*
- cmat bloch2rho (const std::vector< double > &r)

  *Computes the density matrix corresponding to the 3-dimensional real Bloch vector r.*
- template< typename Derived >
  std::size_t hash_eigen (const Eigen::MatrixBase< Derived > &A, std::size_t seed=0)

  *Computes the hash of en Eigen matrix/vector/expression.*
- template< typename Derived >
  internal::IOManipEigen disp (const Eigen::MatrixBase< Derived > &A, double chop=qpp::chop)

  *Eigen expression ostream manipulator.*
- internal::IOManipEigen disp (cplx z, double chop=qpp::chop)

  *Complex number ostream manipulator.*
- template< typename InputIterator >
  internal::IOManipRange< InputIterator > disp (InputIterator first, InputIterator last, const std::string &separator, const std::string &start="[", const std::string &end="]")

  *Range ostream manipulator.*
- template< typename Container >
  internal::IOManipRange< typename Container::const_iterator > disp (const Container &c, const std::string &separator, const std::string &start="[", const std::string &end="]", typename std::enable_if< is_iterable< Container >::value >::type *=nullptr)

  *Standard container ostream manipulator. The container must support std::begin(), std::end() and forward iteration.*
- template< typename PointerType >
  internal::IOManipPointer< PointerType > disp (const PointerType *p, idx N, const std::string &separator, const std::string &start="[", const std::string &end="]")

  *C-style pointer ostream manipulator.*

- template< typename Derived >
  void save (const Eigen::MatrixBase< Derived > &A, const std::string &fname)

  *Saves Eigen expression to a binary file (internal format) in double precision.*
- template< typename Derived >
  dyn_mat< typename Derived::Scalar > load (const std::string &fname)

  *Loads Eigen matrix from a binary file (internal format) in double precision.*
- template< typename Derived >
  dyn_col_vect< typename Derived::Scalar > ip (const Eigen::MatrixBase< Derived > &phi, const Eigen::↩ MatrixBase< Derived > &psi, const std::vector< idx > &subsys, const std::vector< idx > &dims)

  *Generalized inner product.*
- template< typename Derived >
  dyn_col_vect< typename Derived::Scalar > ip (const Eigen::MatrixBase< Derived > &phi, const Eigen::↩ MatrixBase< Derived > &psi, const std::vector< idx > &subsys, idx d=2)

  *Generalized inner product.*
- template< typename Derived >
  std::tuple< idx, std::vector< double >, std::vector< cmat > > measure (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat > &Ks)

  *Measures the state vector or density operator A using the set of Kraus operators Ks.*
- template< typename Derived >
  std::tuple< idx, std::vector< double >, std::vector< cmat > > measure (const Eigen::MatrixBase< Derived > &A, const std::initializer_list< cmat > &Ks)

  *Measures the state vector or density matrix A using the set of Kraus operators Ks.*
- template< typename Derived >
  std::tuple< idx, std::vector< double >, std::vector< cmat > > measure (const Eigen::MatrixBase< Derived > &A, const cmat &U)

  *Measures the state vector or density matrix A in the orthonormal basis specified by the unitary matrix U.*
- template< typename Derived >
  std::tuple< idx, std::vector< double >, std::vector< cmat > > measure (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat > &Ks, const std::vector< idx > &target, const std::vector< idx > &dims)

  *Measures the part subsys of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.*
- template< typename Derived >
  std::tuple< idx, std::vector< double >, std::vector< cmat > > measure (const Eigen::MatrixBase< Derived > &A, const std::initializer_list< cmat > &Ks, const std::vector< idx > &target, const std::vector< idx > &dims)

  *Measures the part target of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.*
- template< typename Derived >
  std::tuple< idx, std::vector< double >, std::vector< cmat > > measure (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat > &Ks, const std::vector< idx > &target, idx d=2)

  *Measures the part target of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.*
- template< typename Derived >
  std::tuple< idx, std::vector< double >, std::vector< cmat > > measure (const Eigen::MatrixBase< Derived > &A, const std::initializer_list< cmat > &Ks, const std::vector< idx > &target, idx d=2)

  *Measures the part target of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.*
- template< typename Derived >
  std::tuple< idx, std::vector< double >, std::vector< cmat > > measure (const Eigen::MatrixBase< Derived > &A, const cmat &V, const std::vector< idx > &target, const std::vector< idx > &dims)

  *Measures the part target of the multi-partite state vector or density matrix A in the orthonormal basis or rank-1 projectors specified by the columns of the matrix V.*
- template< typename Derived >
  std::tuple< idx, std::vector< double >, std::vector< cmat > > measure (const Eigen::MatrixBase< Derived > &A, const cmat &V, const std::vector< idx > &target, idx d=2)

  *Measures the part target of the multi-partite state vector or density matrix A in the orthonormal basis or rank-1 projectors specified by the columns of the matrix V.*

- template< typename Derived >
  std::tuple< std::vector< idx >, double, cmat > measure_seq (const Eigen::MatrixBase< Derived > &A, std::vector< idx > target, std::vector< idx > dims)

  *Sequentially measures the part target of the multi-partite state vector or density matrix A in the computational basis.*
- template< typename Derived >
  std::tuple< std::vector< idx >, double, cmat > measure_seq (const Eigen::MatrixBase< Derived > &A, std::vector< idx > target, idx d=2)

  *Sequentially measures the part target of the multi-partite state vector or density matrix A in the computational basis.*
- template< typename Derived >
  std::enable_if< std::is_same< typename Derived::Scalar, cplx >::value, dyn_mat< cplx > >::type loadMATLAB (const std::string &mat_file, const std::string &var_name)

  *Loads a complex Eigen dynamic matrix from a MATLAB .mat file,.*
- template< typename Derived >
  std::enable_if<!std::is_same< typename Derived::Scalar, cplx >::value, dyn_mat< typename Derived::Scalar > >::type loadMATLAB (const std::string &mat_file, const std::string &var_name)

  *Loads a non-complex Eigen dynamic matrix from a MATLAB .mat file,.*
- template< typename Derived >
  std::enable_if< std::is_same< typename Derived::Scalar, cplx >::value >::type saveMATLAB (const Eigen::MatrixBase< Derived > &A, const std::string &mat_file, const std::string &var_name, const std::string &mode)

  *Saves a complex Eigen dynamic matrix to a MATLAB .mat file,.*
- template< typename Derived >
  std::enable_if< !std::is_same< typename Derived::Scalar, cplx >::value >::type saveMATLAB (const Eigen::MatrixBase< Derived > &A, const std::string &mat_file, const std::string &var_name, const std::string &mode)

  *Saves a non-complex Eigen dynamic matrix to a MATLAB .mat file,.*
- std::vector< int > x2contfrac (double x, idx N, idx cut=1e5)

  *Simple continued fraction expansion.*
- double contfrac2x (const std::vector< int > &cf, idx N=idx(-1))

  *Real representation of a simple continued fraction.*
- bigint gcd (bigint a, bigint b)

  *Greatest common divisor of two integers.*
- bigint gcd (const std::vector< bigint > &as)

  *Greatest common divisor of a list of integers.*
- bigint lcm (bigint a, bigint b)

  *Least common multiple of two integers.*
- bigint lcm (const std::vector< bigint > &as)

  *Least common multiple of a list of integers.*
- std::vector< idx > invperm (const std::vector< idx > &perm)

  *Inverse permutation.*
- std::vector< idx > compperm (const std::vector< idx > &perm, const std::vector< idx > &sigma)

  *Compose permutations.*
- std::vector< bigint > factors (bigint a)

  *Prime factor decomposition.*
- bigint modmul (bigint a, bigint b, bigint p)

  *Modular multiplication without overflow.*
- bigint modpow (bigint a, bigint n, bigint p)

  *Fast integer power modulo p based on the SQUARE-AND-MULTIPLY algorithm.*
- std::tuple< bigint, bigint, bigint > egcd (bigint a, bigint b)

  *Extended greatest common divisor of two integers.*
- bigint modinv (bigint a, bigint p)

  *Modular inverse of a mod p.*

- bool isprime (bigint p, idx k=80)

  *Primality test based on the Miller-Rabin's algorithm.*
- bigint randprime (bigint a, bigint b, idx N=1000)

  *Generates a random big prime uniformly distributed in the interval [a, b].*
- std::vector< std::pair< int, int > > convergents (const std::vector< int > &cf)

  *Convergents.*
- std::vector< std::pair< int, int > > convergents (double x, idx N)

  *Convergents.*
- template<typename Derived1 , typename Derived2 >
  dyn_mat< typename Derived1::Scalar > applyCTRL (const Eigen::MatrixBase< Derived1 > &state, const Eigen::MatrixBase< Derived2 > &A, const std::vector< idx > &ctrl, const std::vector< idx > &target, const std::vector< idx > &dims)

  *Applies the controlled-gate A to the part target of the multi-partite state vector or density matrix state.*
- template<typename Derived1 , typename Derived2 >
  dyn_mat< typename Derived1::Scalar > applyCTRL (const Eigen::MatrixBase< Derived1 > &state, const Eigen::MatrixBase< Derived2 > &A, const std::vector< idx > &ctrl, const std::vector< idx > &target, idx d=2)

  *Applies the controlled-gate A to the part target of the multi-partite state vector or density matrix state.*
- template<typename Derived1 , typename Derived2 >
  dyn_mat< typename Derived1::Scalar > apply (const Eigen::MatrixBase< Derived1 > &state, const Eigen::MatrixBase< Derived2 > &A, const std::vector< idx > &target, const std::vector< idx > &dims)

  *Applies the gate A to the part target of the multi-partite state vector or density matrix state.*
- template<typename Derived1 , typename Derived2 >
  dyn_mat< typename Derived1::Scalar > apply (const Eigen::MatrixBase< Derived1 > &state, const Eigen::MatrixBase< Derived2 > &A, const std::vector< idx > &target, idx d=2)

  *Applies the gate A to the part target of the multi-partite state vector or density matrix state.*
- template<typename Derived >
  cmat apply (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat > &Ks)

  *Applies the channel specified by the set of Kraus operators Ks to the density matrix A.*
- template<typename Derived >
  cmat apply (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat > &Ks, const std::vector< idx > &target, const std::vector< idx > &dims)

  *Applies the channel specified by the set of Kraus operators Ks to the part target of the multi-partite density matrix A.*
- template<typename Derived >
  cmat apply (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat > &Ks, const std::vector< idx > &target, idx d=2)

  *Applies the channel specified by the set of Kraus operators Ks to the part target of the multi-partite density matrix A.*
- cmat kraus2super (const std::vector< cmat > &Ks)

  *Superoperator matrix.*
- cmat kraus2choi (const std::vector< cmat > &Ks)

  *Choi matrix.*
- std::vector< cmat > choi2kraus (const cmat &A)

  *Orthogonal Kraus operators from Choi matrix.*
- cmat choi2super (const cmat &A)

  *Converts Choi matrix to superoperator matrix.*
- cmat super2choi (const cmat &A)

  *Converts superoperator matrix to Choi matrix.*
- template<typename Derived >
  dyn_mat< typename Derived::Scalar > ptrace1 (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)

  *Partial trace.*
- template<typename Derived >
  dyn_mat< typename Derived::Scalar > ptrace1 (const Eigen::MatrixBase< Derived > &A, idx d=2)

*Partial trace.*

- template< typename Derived >
dyn_mat< typename Derived::Scalar > ptrace2 (const Eigen::MatrixBase< Derived > &A, const std↩ ::vector< idx > &dims)

  *Partial trace.*
- template< typename Derived >
dyn_mat< typename Derived::Scalar > ptrace2 (const Eigen::MatrixBase< Derived > &A, idx d=2)

  *Partial trace.*
- template< typename Derived >
dyn_mat< typename Derived::Scalar > ptrace (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &target, const std::vector< idx > &dims)

  *Partial trace.*
- template< typename Derived >
dyn_mat< typename Derived::Scalar > ptrace (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &target, idx d=2)

  *Partial trace.*
- template< typename Derived >
dyn_mat< typename Derived::Scalar > ptranspose (const Eigen::MatrixBase< Derived > &A, const std↩ ::vector< idx > &target, const std::vector< idx > &dims)

  *Partial transpose.*
- template< typename Derived >
dyn_mat< typename Derived::Scalar > ptranspose (const Eigen::MatrixBase< Derived > &A, const std↩ ::vector< idx > &target, idx d=2)

  *Partial transpose.*
- template< typename Derived >
dyn_mat< typename Derived::Scalar > syspermute (const Eigen::MatrixBase< Derived > &A, const std↩ ::vector< idx > &perm, const std::vector< idx > &dims)

  *Subsystem permutation.*
- template< typename Derived >
dyn_mat< typename Derived::Scalar > syspermute (const Eigen::MatrixBase< Derived > &A, const std↩ ::vector< idx > &perm, idx d=2)

  *Subsystem permutation.*
- template< typename Derived >
dyn_mat< typename Derived::Scalar > applyQFT (const Eigen::MatrixBase< Derived > &A, const std↩ ::vector< idx > &target, idx d=2, bool swap=true)

  *Applies the qudit quantum Fourier transform to the part target of the multi-partite state vector or density matrix A.*
- template< typename Derived >
dyn_mat< typename Derived::Scalar > applyTFQ (const Eigen::MatrixBase< Derived > &A, const std↩ ::vector< idx > &target, idx d=2, bool swap=true)

  *Applies the inverse (adjoint) qudit quantum Fourier transform to the part target of the multi-partite state vector or density matrix A.*
- template< typename Derived >
dyn_col_vect< typename Derived::Scalar > TFQ (const Eigen::MatrixBase< Derived > &A, idx d=2, bool swap=true)

  *Inverse (adjoint) qudit quantum Fourier transform.*
- template< typename Derived >
dyn_col_vect< typename Derived::Scalar > QFT (const Eigen::MatrixBase< Derived > &A, idx d=2, bool swap=true)

  *Qudit quantum Fourier transform.*
- double rand (double a, double b)

  *Generates a random real number uniformly distributed in the interval [a, b)*
- bigint rand (bigint a, bigint b)

  *Generates a random big integer uniformly distributed in the interval [a, b].*
- idx randidx (idx a=std::numeric_limits< idx >::min(), idx b=std::numeric_limits< idx >::max())

*Generates a random index (idx) uniformly distributed in the interval [a, b].*

- template< typename Derived >
  Derived rand (idx rows QPP_UNUSED_, idx cols QPP_UNUSED_, double a QPP_UNUSED_=0, double b QPP_UNUSED_=1)

  *Generates a random matrix with entries uniformly distributed in the interval [a, b)*
- template<>
  dmat rand (idx rows, idx cols, double a, double b)

  *Generates a random real matrix with entries uniformly distributed in the interval [a, b), specialization for double matrices (qpp::dmat)*
- template<>
  cmat rand (idx rows, idx cols, double a, double b)

  *Generates a random complex matrix with entries (both real and imaginary) uniformly distributed in the interval [a, b), specialization for complex matrices (qpp::cmat)*
- template< typename Derived >
  Derived randn (idx rows QPP_UNUSED_, idx cols QPP_UNUSED_, double mean QPP_UNUSED_=0, double sigma QPP_UNUSED_=1)

  *Generates a random matrix with entries normally distributed in N(mean, sigma)*
- template<>
  dmat randn (idx rows, idx cols, double mean, double sigma)

  *Generates a random real matrix with entries normally distributed in N(mean, sigma), specialization for double matrices (qpp::dmat)*
- template<>
  cmat randn (idx rows, idx cols, double mean, double sigma)

  *Generates a random complex matrix with entries (both real and imaginary) normally distributed in N(mean, sigma), specialization for complex matrices (qpp::cmat)*
- double randn (double mean=0, double sigma=1)

  *Generates a random real number (double) normally distributed in N(mean, sigma)*
- cmat randU (idx D=2)

  *Generates a random unitary matrix.*
- cmat randV (idx Din, idx Dout)

  *Generates a random isometry matrix.*
- std::vector< cmat > randkraus (idx N, idx D=2)

  *Generates a set of random Kraus operators.*
- cmat randH (idx D=2)

  *Generates a random Hermitian matrix.*
- ket randket (idx D=2)

  *Generates a random normalized ket (pure state vector)*
- cmat randrho (idx D=2)

  *Generates a random density matrix.*
- std::vector< idx > randperm (idx N)

  *Generates a random uniformly distributed permutation.*
- std::vector< double > randprob (idx N)

  *Generates a random probability vector uniformly distributed over the probability simplex.*
- std::vector< double > uniform (idx N)

  *Uniform probability distribution vector.*
- std::vector< double > marginalX (const dmat &probXY)

  *Marginal distribution.*
- std::vector< double > marginalY (const dmat &probXY)

  *Marginal distribution.*
- template< typename Container >
  double avg (const std::vector< double > &prob, const Container &X, typename std::enable_if< is_iterable< Container >::value >::type *=nullptr)

  *Average.*

- template< typename Container >
  double cov (const dmat &probXY, const Container &X, const Container &Y, typename std::enable_if< is_iterable< Container >::value >::type ∗=nullptr)

  *Covariance.*
- template< typename Container >
  double var (const std::vector< double > &prob, const Container &X, typename std::enable_if< is_iterable< Container >::value >::type ∗=nullptr)

  *Variance.*
- template< typename Container >
  double sigma (const std::vector< double > &prob, const Container &X, typename std::enable_if< is_iterable< Container >::value >::type ∗=nullptr)

  *Standard deviation.*
- template< typename Container >
  double cor (const dmat &probXY, const Container &X, const Container &Y, typename std::enable_if< is_iterable< Container >::value >::type ∗=nullptr)

  *Correlation.*

## Variables

- constexpr double chop = 1e-10

  *Used in qpp::disp() for setting to zero numbers that have their absolute value smaller than qpp::chop.*
- constexpr idx maxn = 64

  *Maximum number of allowed qubits/qudits (subsystems)*
- constexpr double pi = 3.141592653589793238462643383279502884

  $\pi$
- constexpr double ee = 2.718281828459045235360287471352662497

  *Base of natural logarithm,* $e$.
- constexpr double infty = std::numeric_limits<double>::max()

  *Used to denote infinity in double precision.*

### 6.1.1 Detailed Description

Quantum++ main namespace.

### 6.1.2 Typedef Documentation

#### 6.1.2.1 bigint

```
using qpp::bigint = typedef long long int
```

Big integer.

#### 6.1.2.2 bra

```
using qpp::bra = typedef Eigen::RowVectorXcd
```

Complex (double precision) dynamic Eigen row vector.

#### 6.1.2.3 cmat

```
using qpp::cmat = typedef Eigen::MatrixXcd
```

Complex (double precision) dynamic Eigen matrix.

#### 6.1.2.4 cplx

```
using qpp::cplx = typedef std::complex<double>
```

Complex number in double precision.

#### 6.1.2.5 dmat

```
using qpp::dmat = typedef Eigen::MatrixXd
```

Real (double precision) dynamic Eigen matrix.

#### 6.1.2.6 dyn_col_vect

```
template<typename Scalar >
using qpp::dyn_col_vect = typedef Eigen::Matrix<Scalar, Eigen::Dynamic, 1>
```

Dynamic Eigen column vector over the field specified by *Scalar*.

Example:

```
// type of colvect is Eigen::Matrix<float, Eigen::Dynamic, 1>
dyn_col_vect<float> colvect(2);
```

#### 6.1.2.7 dyn_mat

```
template<typename Scalar >
using qpp::dyn_mat = typedef Eigen::Matrix<Scalar, Eigen::Dynamic, Eigen::Dynamic>
```

Dynamic Eigen matrix over the field specified by *Scalar*.

Example:

```
// type of mat is Eigen::Matrix<float, Eigen::Dynamic, Eigen::Dynamic>
dyn_mat<float> mat(2, 3);
```

#### 6.1.2.8 dyn_row_vect

```
template<typename Scalar >
using qpp::dyn_row_vect = typedef Eigen::Matrix<Scalar, 1, Eigen::Dynamic>
```

Dynamic Eigen row vector over the field specified by *Scalar*.

Example:

```
// type of rowvect is Eigen::Matrix<float, 1, Eigen::Dynamic>
dyn_row_vect<float> rowvect(3);
```

#### 6.1.2.9 idx

```
using qpp::idx = typedef std::size_t
```

Non-negative integer index, make sure you use an unsigned type.

#### 6.1.2.10 ket

```
using qpp::ket = typedef Eigen::VectorXcd
```

Complex (double precision) dynamic Eigen column vector.

#### 6.1.2.11 to_void

```
template<typename... Ts>
using qpp::to_void = typedef typename make_void<Ts...>::type
```

Alias template that implements the proposal for void_t.

**See also**

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2014/n3911

### 6.1.3 Function Documentation

#### 6.1.3.1 absm()

```
template<typename Derived >
cmat qpp::absm (
            const Eigen::MatrixBase< Derived > & A )
```

Matrix absolute value.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Matrix absolute value of *A*

#### 6.1.3.2 abssq() [1/3]

```
template<typename InputIterator >
std::vector<double> qpp::abssq (
            InputIterator first,
            InputIterator last )
```

Computes the absolute values squared of an STL-like range of complex numbers.

**Parameters**

| | |
|---|---|
| *first* | Iterator to the first element of the range |
| *last* | Iterator to the last element of the range |

**Returns**

Real vector consisting of the range absolute values squared

#### 6.1.3.3 abssq() [2/3]

```
template<typename Container >
std::vector<double> qpp::abssq (
            const Container & c,
            typename std::enable_if< is_iterable< Container >::value >::type * = nullptr )
```

Computes the absolute values squared of an STL-like container.

**Parameters**

| | |
|---|---|
| *c* | STL-like container |

**Returns**

Real vector consisting of the container's absolute values squared

#### 6.1.3.4 abssq() [3/3]

```
template<typename Derived >
std::vector<double> qpp::abssq (
            const Eigen::MatrixBase< Derived > & A )
```

Computes the absolute values squared of an Eigen expression.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Real vector consisting of the absolute values squared

#### 6.1.3.5 adjoint()

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::adjoint (
            const Eigen::MatrixBase< Derived > & A )
```

Adjoint.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Adjoint (Hermitian conjugate) of *A*, as a dynamic matrix over the same scalar field as *A*

#### 6.1.3.6 anticomm()

```
template<typename Derived1 , typename Derived2 >
dyn_mat<typename Derived1::Scalar> qpp::anticomm (
            const Eigen::MatrixBase< Derived1 > & A,
            const Eigen::MatrixBase< Derived2 > & B )
```

Anti-commutator.

**See also**

qpp::comm()

Anti-commutator $\{A, B\} = AB + BA$. Both *A* and *B* must be Eigen expressions over the same scalar field.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *B* | Eigen expression |

**Returns**

Anti-commutator $AB + BA$, as a dynamic matrix over the same scalar field as *A*

#### 6.1.3.7 apply() [1/5]

```
template<typename Derived1 , typename Derived2 >
dyn_mat<typename Derived1::Scalar> qpp::apply (
            const Eigen::MatrixBase< Derived1 > & state,
            const Eigen::MatrixBase< Derived2 > & A,
            const std::vector< idx > & target,
            const std::vector< idx > & dims )
```

Applies the gate *A* to the part *target* of the multi-partite state vector or density matrix *state*.

**Note**

The dimension of the gate *A* must match the dimension of *target*

**Parameters**

| | |
|---|---|
| *state* | Eigen expression |
| *A* | Eigen expression |
| *target* | Subsystem indexes where the gate *A* is applied |
| *dims* | Dimensions of the multi-partite system |

**Returns**

Gate *A* applied to the part *target* of *state*

#### 6.1.3.8 apply() [2/5]

```
template<typename Derived1 , typename Derived2 >
dyn_mat<typename Derived1::Scalar> qpp::apply (
            const Eigen::MatrixBase< Derived1 > & state,
            const Eigen::MatrixBase< Derived2 > & A,
            const std::vector< idx > & target,
            idx d = 2 )
```

Applies the gate *A* to the part *target* of the multi-partite state vector or density matrix *state*.

**Note**

The dimension of the gate *A* must match the dimension of *target*

**Parameters**

| | |
|---|---|
| *state* | Eigen expression |
| *A* | Eigen expression |
| *target* | Subsystem indexes where the gate *A* is applied |
| *d* | Subsystem dimensions |

**Returns**

Gate *A* applied to the part *target* of *state*

#### 6.1.3.9 apply() [3/5]

```
template<typename Derived >
cmat qpp::apply (
            const Eigen::MatrixBase< Derived > & A,
            const std::vector< cmat > & Ks )
```

Applies the channel specified by the set of Kraus operators *Ks* to the density matrix *A*.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *Ks* | Set of Kraus operators |

**Returns**

Output density matrix after the action of the channel

#### 6.1.3.10 apply() [4/5]

```
template<typename Derived >
cmat qpp::apply (
            const Eigen::MatrixBase< Derived > & A,
            const std::vector< cmat > & Ks,
            const std::vector< idx > & target,
            const std::vector< idx > & dims )
```

Applies the channel specified by the set of Kraus operators *Ks* to the part *target* of the multi-partite density matrix *A*.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *Ks* | Set of Kraus operators |
| *target* | Subsystem indexes where the Kraus operators *Ks* are applied |
| *dims* | Dimensions of the multi-partite system |

**Returns**

Output density matrix after the action of the channel

#### 6.1.3.11 apply() [5/5]

```
template<typename Derived >
cmat qpp::apply (
            const Eigen::MatrixBase< Derived > & A,
            const std::vector< cmat > & Ks,
            const std::vector< idx > & target,
            idx d = 2 )
```

Applies the channel specified by the set of Kraus operators *Ks* to the part *target* of the multi-partite density matrix *A*.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *Ks* | Set of Kraus operators |
| *target* | Subsystem indexes where the Kraus operators *Ks* are applied |
| *d* | Subsystem dimensions |

**Returns**

Output density matrix after the action of the channel

#### 6.1.3.12 applyCTRL() [1/2]

```
template<typename Derived1 , typename Derived2 >
dyn_mat<typename Derived1::Scalar> qpp::applyCTRL (
            const Eigen::MatrixBase< Derived1 > & state,
            const Eigen::MatrixBase< Derived2 > & A,
            const std::vector< idx > & ctrl,
            const std::vector< idx > & target,
            const std::vector< idx > & dims )
```

Applies the controlled-gate *A* to the part *target* of the multi-partite state vector or density matrix *state*.

**See also**

qpp::Gates::CTRL()

**Note**

The dimension of the gate *A* must match the dimension of *target*. Also, all control subsystems in *ctrl* must have the same dimension.

**Parameters**

| | |
|---|---|
| *state* | Eigen expression |
| *A* | Eigen expression |
| *ctrl* | Control subsystem indexes |
| *target* | Subsystem indexes where the gate *A* is applied |
| *dims* | Dimensions of the multi-partite system |

**Returns**

CTRL-A gate applied to the part *target* of *state*

#### 6.1.3.13 applyCTRL() [2/2]

```
template<typename Derived1 , typename Derived2 >
dyn_mat<typename Derived1::Scalar> qpp::applyCTRL (
            const Eigen::MatrixBase< Derived1 > & state,
            const Eigen::MatrixBase< Derived2 > & A,
            const std::vector< idx > & ctrl,
            const std::vector< idx > & target,
            idx d = 2 )
```

Applies the controlled-gate *A* to the part *target* of the multi-partite state vector or density matrix *state*.

**See also**

qpp::Gates::CTRL()

**Note**

The dimension of the gate *A* must match the dimension of *target*

**Parameters**

| | |
|---|---|
| *state* | Eigen expression |
| *A* | Eigen expression |
| *ctrl* | Control subsystem indexes |
| *target* | Subsystem indexes where the gate *A* is applied |
| *d* | Subsystem dimensions |

**Returns**

CTRL-A gate applied to the part *target* of *state*

#### 6.1.3.14 applyQFT()

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::applyQFT (
            const Eigen::MatrixBase< Derived > & A,
            const std::vector< idx > & target,
            idx d = 2,
            bool swap = true )
```

Applies the qudit quantum Fourier transform to the part *target* of the multi-partite state vector or density matrix *A*.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *target* | Subsystem indexes where the QFT is applied |
| *d* | Subsystem dimensions |
| *swap* | Swaps the qubits/qudits at the end (true by default) |

**Returns**

Qudit Quantum Fourier transform applied to the part *target* of *A*

#### 6.1.3.15 applyTFQ()

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::applyTFQ (
            const Eigen::MatrixBase< Derived > & A,
            const std::vector< idx > & target,
            idx d = 2,
            bool swap = true )
```

Applies the inverse (adjoint) qudit quantum Fourier transform to the part *target* of the multi-partite state vector or density matrix *A*.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *target* | Subsystem indexes where the TFQ is applied |
| *d* | Subsystem dimensions |
| *swap* | Swaps the qubits/qudits at the end (true by default) |

**Returns**

Inverse (adjoint) qudit Quantum Fourier transform applied to the part *target* of *A*

#### 6.1.3.16 avg()

```
template<typename Container >
double qpp::avg (
            const std::vector< double > & prob,
            const Container & X,
            typename std::enable_if< is_iterable< Container >::value >::type * = nullptr )
```

Average.

**Parameters**

| | |
|---|---|
| *prob* | Real probability vector representing the probability distribution of *X* |
| *X* | Real random variable values represented by an STL-like container |

**Returns**

Average of *X*

#### 6.1.3.17 bloch2rho()

```
cmat qpp::bloch2rho (
            const std::vector< double > & r ) [inline]
```

Computes the density matrix corresponding to the 3-dimensional real Bloch vector *r*.

**See also**

qpp::rho2bloch()

**Parameters**

| | |
|---|---|
| *r* | 3-dimensional real vector |

**Returns**

Qubit density matrix

#### 6.1.3.18 choi2kraus()

```
std::vector<cmat> qpp::choi2kraus (
            const cmat & A ) [inline]
```

Orthogonal Kraus operators from Choi matrix.

**See also**

qpp::kraus2choi()

Extracts a set of orthogonal (under Hilbert-Schmidt operator norm) Kraus operators from the Choi matrix *A*

**Note**

The Kraus operators satisfy $Tr(K_i^\dagger K_j) = \delta_{ij}$ for all $i \neq j$

**Parameters**

| | |
|---|---|
| *A* | Choi matrix |

**Returns**

Set of orthogonal Kraus operators

#### 6.1.3.19 choi2super()

```
cmat qpp::choi2super (
            const cmat & A )  [inline]
```

Converts Choi matrix to superoperator matrix.

**See also**

qpp::super2choi()

**Parameters**

| | |
|---|---|
| *A* | Choi matrix |

**Returns**

Superoperator matrix

#### 6.1.3.20 comm()

```
template<typename Derived1 , typename Derived2 >
dyn_mat<typename Derived1::Scalar> qpp::comm (
            const Eigen::MatrixBase< Derived1 > & A,
            const Eigen::MatrixBase< Derived2 > & B )
```

Commutator.

**See also**

qpp::anticomm()

Commutator $[A, B] = AB - BA$. Both *A* and *B* must be Eigen expressions over the same scalar field.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *B* | Eigen expression |

**Returns**

Commutator $AB - BA$, as a dynamic matrix over the same scalar field as *A*

#### 6.1.3.21 complement()

```
std::vector<idx> qpp::complement (
            std::vector< idx > subsys,
            idx n ) [inline]
```

Constructs the complement of a subsystem vector.

**Parameters**

| | |
|---|---|
| *subsys* | Subsystem vector |
| *n* | Total number of systems |

**Returns**

Complement of *subsys* with respect to the set $\{0, 1, \ldots, n-1\}$

#### 6.1.3.22 compperm()

```
std::vector<idx> qpp::compperm (
            const std::vector< idx > & perm,
            const std::vector< idx > & sigma ) [inline]
```

Compose permutations.

**Parameters**

| | |
|---|---|
| *perm* | Permutation |
| *sigma* | Permutation |

**Returns**

Composition of the permutations *perm* ∘ *sigma* = perm(sigma)

#### 6.1.3.23 concurrence()

```
template<typename Derived >
double qpp::concurrence (
            const Eigen::MatrixBase< Derived > & A )
```

Wootters concurrence of the bi-partite qubit mixed state *A*.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Wootters concurrence

#### 6.1.3.24 conjugate()

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::conjugate (
            const Eigen::MatrixBase< Derived > & A )
```

Complex conjugate.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Complex conjugate of *A*, as a dynamic matrix over the same scalar field as *A*

#### 6.1.3.25 contfrac2x()

```
double qpp::contfrac2x (
            const std::vector< int > & cf,
            idx N = idx(-1) )  [inline]
```

Real representation of a simple continued fraction.

**See also**

qpp::x2contfrac()

**Note**

If *N* is greater than the size of *cf* (by default it is), then all terms in *cf* are considered.

**Parameters**

| | |
|---|---|
| *cf* | Integer vector containing the simple continued fraction expansion |
| *N* | Number of terms considered in the continued fraction expansion. |

**Returns**

Real representation of the simple continued fraction

#### 6.1.3.26 convergents() [1/2]

```
std::vector<std::pair<int, int> > qpp::convergents (
            const std::vector< int > & cf ) [inline]
```

Convergents.

**See also**

qpp::contfrac2x() and qpp::x2contfrac()

**Parameters**

| | |
|---|---|
| *cf* | Continued fraction |

**Returns**

Vector of convergents pairs $(a_k, b_k)$ that approximate the number represented by the continued fraction

#### 6.1.3.27 convergents() [2/2]

```
std::vector<std::pair<int, int> > qpp::convergents (
            double x,
            idx N ) [inline]
```

Convergents.

**See also**

qpp::contfrac2x() and qpp::x2contfrac()

**Note**

In the continued fraction expansion of *x* has less terms than *N*, then the series of convergents is truncated to the number of terms in the continued fraction expansion of *x*.

**Parameters**

| | |
|---|---|
| *x* | Real number |
| *N* | Number of convergents. |

**Returns**

Vector of convergents pairs $(a_k, b_k)$ that approximate the number *x*

#### 6.1.3.28 cor()

```
template<typename Container >
double qpp::cor (
            const dmat & probXY,
            const Container & X,
            const Container & Y,
            typename std::enable_if< is_iterable< Container >::value >::type * = nullptr )
```

Correlation.

**Parameters**

| | |
|---|---|
| *probXY* | Real matrix representing the joint probability distribution of *X* and *Y* in lexicographical order (*X* labels the rows, *Y* labels the columns) |
| *X* | Real random variable values represented by an STL-like container |
| *Y* | Real random variable values represented by an STL-like container |

**Returns**

Correlation of *X* and *Y*

#### 6.1.3.29 cosm()

```
template<typename Derived >
cmat qpp::cosm (
            const Eigen::MatrixBase< Derived > & A )
```

Matrix cos.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Matrix cosine of *A*

#### 6.1.3.30 cov()

```
template<typename Container >
double qpp::cov (
            const dmat & probXY,
            const Container & X,
            const Container & Y,
            typename std::enable_if< is_iterable< Container >::value >::type * = nullptr )
```

Covariance.

**Parameters**

| | |
|---|---|
| *probXY* | Real matrix representing the joint probability distribution of *X* and *Y* in lexicographical order (*X* labels the rows, *Y* labels the columns) |
| *X* | Real random variable values represented by an STL-like container |
| *Y* | Real random variable values represented by an STL-like container |

**Returns**

Covariance of *X* and *Y*

#### 6.1.3.31 cwise()

```
template<typename OutputScalar , typename Derived >
dyn_mat<OutputScalar> qpp::cwise (
            const Eigen::MatrixBase< Derived > & A,
            OutputScalar(*)(const typename Derived::Scalar &) f )
```

Functor.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *f* | Pointer-to-function from scalars of *A* to *OutputScalar* |

**Returns**

Component-wise $f(A)$, as a dynamic matrix over the *OutputScalar* scalar field

#### 6.1.3.32 det()

```
template<typename Derived >
Derived::Scalar qpp::det (
            const Eigen::MatrixBase< Derived > & A )
```

Determinant.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Determinant of *A*, as a scalar over the same scalar field as *A*. Returns $\pm\infty$ when the determinant overflows/underflows.

#### 6.1.3.33 dirsum() [1/4]

```
template<typename T >
dyn_mat<typename T::Scalar> qpp::dirsum (
            const T & head )
```

Direct sum.

**See also**

qpp::dirsumpow()

Used to stop the recursion for the variadic template version of qpp::dirsum()

**Parameters**

| | |
|---|---|
| *head* | Eigen expression |

**Returns**

Its argument *head*

#### 6.1.3.34 dirsum() [2/4]

```
template<typename T , typename... Args>
dyn_mat<typename T::Scalar> qpp::dirsum (
            const T & head,
            const Args &... tail )
```

Direct sum.

**See also**

qpp::dirsumpow()

**Parameters**

| | |
|---|---|
| *head* | Eigen expression |
| *tail* | Variadic Eigen expression (zero or more parameters) |

**Returns**

Direct sum of all input parameters, evaluated from left to right, as a dynamic matrix over the same scalar field as its arguments

#### 6.1.3.35 dirsum() [3/4]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::dirsum (
            const std::vector< Derived > & As )
```

Direct sum.

**See also**

qpp::dirsumpow()

**Parameters**

| | |
|---|---|
| *As* | std::vector of Eigen expressions |

**Returns**

Direct sum of all elements in *As*, evaluated from left to right, as a dynamic matrix over the same scalar field as its arguments

#### 6.1.3.36 dirsum() [4/4]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::dirsum (
            const std::initializer_list< Derived > & As )
```

Direct sum.

**See also**

qpp::dirsumpow()

**Parameters**

| | |
|---|---|
| *As* | std::initializer_list of Eigen expressions, such as *{A1*, A2, ... ,Ak} |

**Returns**

Direct sum of all elements in *As*, evaluated from left to right, as a dynamic matrix over the same scalar field as its arguments

#### 6.1.3.37 dirsumpow()

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::dirsumpow (
```

```
            const Eigen::MatrixBase< Derived > & A,
            idx n )
```

Direct sum power.

**See also**

qpp::dirsum()

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *n* | Non-negative integer |

**Returns**

Direct sum of *A* with itself *n* times $A^{\oplus n}$, as a dynamic matrix over the same scalar field as *A*

#### 6.1.3.38 disp() [1/5]

```
template<typename Derived >
internal::IOManipEigen qpp::disp (
            const Eigen::MatrixBase< Derived > & A,
            double chop = qpp::chop )
```

Eigen expression ostream manipulator.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *chop* | Set to zero the elements smaller in absolute value than *chop* |

**Returns**

Instance of qpp::internal::IOManipEigen

#### 6.1.3.39 disp() [2/5]

```
internal::IOManipEigen qpp::disp (
            cplx z,
            double chop = qpp::chop ) [inline]
```

Complex number ostream manipulator.

**Parameters**

| | |
|---|---|
| *z* | Complex number (or any other type implicitly cast-able to std::complex<double>) |
| *chop* | Set to zero the elements smaller in absolute value than *chop* |

**Returns**

Instance of qpp::internal::IOManipEigen

#### 6.1.3.40 disp() [3/5]

```
template<typename InputIterator >
internal::IOManipRange<InputIterator> qpp::disp (
            InputIterator first,
            InputIterator last,
            const std::string & separator,
            const std::string & start = "[",
            const std::string & end = "]" )
```

Range ostream manipulator.

**Parameters**

| | |
|---|---|
| *first* | Iterator to the first element of the range |
| *last* | Iterator to the last element of the range |
| *separator* | Separator |
| *start* | Left marking |
| *end* | Right marking |

**Returns**

Instance of qpp::internal::IOManipRange

#### 6.1.3.41 disp() [4/5]

```
template<typename Container >
internal::IOManipRange<typename Container::const_iterator> qpp::disp (
            const Container & c,
            const std::string & separator,
            const std::string & start = "[",
            const std::string & end = "]",
            typename std::enable_if< is_iterable< Container >::value >::type * = nullptr )
```

Standard container ostream manipulator. The container must support std::begin(), std::end() and forward iteration.

**Parameters**

| | |
|---|---|
| *c* | Container |
| *separator* | Separator |
| *start* | Left marking |
| *end* | Right marking |

**Returns**

Instance of qpp::internal::IOManipRange

#### 6.1.3.42 disp() [5/5]

```
template<typename PointerType >
internal::IOManipPointer<PointerType> qpp::disp (
            const PointerType * p,
            idx N,
            const std::string & separator,
            const std::string & start = "[",
            const std::string & end = "]" )
```

C-style pointer ostream manipulator.

**Parameters**

| | |
|---|---|
| *p* | Pointer to the first element |
| *N* | Number of elements to be displayed |
| *separator* | Separator |
| *start* | Left marking |
| *end* | Right marking |

**Returns**

Instance of qpp::internal::IOManipPointer

#### 6.1.3.43 egcd()

```
std::tuple<bigint, bigint, bigint> qpp::egcd (
            bigint a,
            bigint b )  [inline]
```

Extended greatest common divisor of two integers.

**See also**

qpp::gcd()

**Parameters**

| | |
|---|---|
| *a* | Integer |
| *b* | Integer |

**Returns**

Tuple of: 1. Integer $m$, 2. Integer $n$, and 3. Non-negative integer $gcd(a, b)$ such that $ma + nb = gcd(a, b)$

#### 6.1.3.44 eig()

```
template<typename Derived >
std::pair<dyn_col_vect<cplx>, cmat> qpp::eig (
            const Eigen::MatrixBase< Derived > & A )
```

Full eigen decomposition.

**See also**

qpp::heig()

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Pair of: 1. Eigenvalues of *A*, as a complex dynamic column vector, and 2. Eigenvectors of *A*, as columns of a complex dynamic matrix

#### 6.1.3.45 entanglement() [1/2]

```
template<typename Derived >
double qpp::entanglement (
            const Eigen::MatrixBase< Derived > & A,
            const std::vector< idx > & dims )
```

Entanglement of the bi-partite pure state *A*.

Defined as the von-Neumann entropy of the reduced density matrix of one of the subsystems

**See also**

qpp::entropy()

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *dims* | Dimensions of the bi-partite system |

**Returns**

Entanglement, with the logarithm in base 2

#### 6.1.3.46 entanglement() [2/2]

```
template<typename Derived >
double qpp::entanglement (
            const Eigen::MatrixBase< Derived > & A,
            idx d = 2 )
```

Entanglement of the bi-partite pure state *A*.

Defined as the von-Neumann entropy of the reduced density matrix of one of the subsystems

**See also**

qpp::entropy()

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *d* | Subsystem dimensions |

**Returns**

Entanglement, with the logarithm in base 2

#### 6.1.3.47 entropy() [1/2]

```
template<typename Derived >
double qpp::entropy (
            const Eigen::MatrixBase< Derived > & A )
```

von-Neumann entropy of the density matrix *A*

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

von-Neumann entropy, with the logarithm in base 2

#### 6.1.3.48 entropy() [2/2]

```
double qpp::entropy (
            const std::vector< double > & prob ) [inline]
```

Shannon entropy of the probability distribution *prob*.

**Parameters**

| | |
|---|---|
| *prob* | Real probability vector |

**Returns**

Shannon entropy, with the logarithm in base 2

#### 6.1.3.49 evals()

```
template<typename Derived >
dyn_col_vect<cplx> qpp::evals (
            const Eigen::MatrixBase< Derived > & A )
```

Eigenvalues.

**See also**

qpp::hevals()

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Eigenvalues of *A*, as a complex dynamic column vector

#### 6.1.3.50 evects()

```
template<typename Derived >
cmat qpp::evects (
            const Eigen::MatrixBase< Derived > & A )
```

Eigenvectors.

**See also**

qpp::hevects()

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Eigenvectors of *A*, as columns of a complex dynamic matrix

#### 6.1.3.51 expm()

```
template<typename Derived >
cmat qpp::expm (
            const Eigen::MatrixBase< Derived > & A )
```

Matrix exponential.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Matrix exponential of *A*

#### 6.1.3.52 factors()

```
std::vector<bigint> qpp::factors (
            bigint a )  [inline]
```

Prime factor decomposition.

**Note**

Runs in $\mathcal{O}(\sqrt{n})$ time complexity

**Parameters**

| | |
|---|---|
| *a* | Integer different from 0, 1 or -1 |

**Returns**

Integer vector containing the factors

#### 6.1.3.53 funm()

```
template<typename Derived >
cmat qpp::funm (
            const Eigen::MatrixBase< Derived > & A,
            cplx(*)(const cplx &) f )
```

Functional calculus f(A)

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *f* | Pointer-to-function from complex to complex |

**Returns**

$f(A)$

#### 6.1.3.54 gcd() [1/2]

```
bigint qpp::gcd (
            bigint a,
            bigint b ) [inline]
```

Greatest common divisor of two integers.

**See also**

qpp::lcm()

**Parameters**

| | |
|---|---|
| *a* | Integer |
| *b* | Integer |

**Returns**

Greatest common divisor of *a* and *b*

#### 6.1.3.55 gcd() [2/2]

```
bigint qpp::gcd (
            const std::vector< bigint > & as )  [inline]
```

Greatest common divisor of a list of integers.

**See also**

qpp::lcm()

**Parameters**

| | |
|---|---|
| *as* | List of integers |

**Returns**

Greatest common divisor of all numbers in *as*

#### 6.1.3.56 gconcurrence()

```
template<typename Derived >
double qpp::gconcurrence (
            const Eigen::MatrixBase< Derived > & A )
```

G-concurrence of the bi-partite pure state *A*.

**Note**

Both local dimensions must be equal

Uses qpp::logdet() to avoid overflows

**See also**

qpp::logdet()

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

G-concurrence

#### 6.1.3.57 grams() [1/3]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::grams (
            const std::vector< Derived > & As )
```

Gram-Schmidt orthogonalization.

**Parameters**

| | |
|---|---|
| *As* | std::vector of Eigen expressions as column vectors |

**Returns**

Gram-Schmidt vectors of *As* as columns of a dynamic matrix over the same scalar field as its arguments

#### 6.1.3.58 grams() [2/3]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::grams (
            const std::initializer_list< Derived > & As )
```

Gram-Schmidt orthogonalization.

**Parameters**

| | |
|---|---|
| *As* | std::initializer_list of Eigen expressions as column vectors |

**Returns**

Gram-Schmidt vectors of *As* as columns of a dynamic matrix over the same scalar field as its arguments

#### 6.1.3.59 grams() [3/3]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::grams (
            const Eigen::MatrixBase< Derived > & A )
```

Gram-Schmidt orthogonalization.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression, the input vectors are the columns of *A* |

**Returns**

Gram-Schmidt vectors of the columns of *A*, as columns of a dynamic matrix over the same scalar field as *A*

#### 6.1.3.60 hash_eigen()

```
template<typename Derived >
std::size_t qpp::hash_eigen (
            const Eigen::MatrixBase< Derived > & A,
            std::size_t seed = 0 )
```

Computes the hash of en Eigen matrix/vector/expression.

**Note**

Code taken from boost::hash_combine(), see https://www.boost.org/doc/libs/1_69_0/doc/html/hash/reference.html#boost.hash_combine

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *seed* | Seed, 0 by default |

**Returns**

Hash of its argument

#### 6.1.3.61 heig()

```
template<typename Derived >
std::pair<dyn_col_vect<double>, cmat> qpp::heig (
            const Eigen::MatrixBase< Derived > & A )
```

Full eigen decomposition of Hermitian expression.

**See also**

qpp::eig()

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Pair of: 1. Eigenvalues of *A*, as a real dynamic column vector, and 2. Eigenvectors of *A*, as columns of a complex dynamic matrix

#### 6.1.3.62 hevals()

```
template<typename Derived >
dyn_col_vect<double> qpp::hevals (
            const Eigen::MatrixBase< Derived > & A )
```

Hermitian eigenvalues.

**See also**

qpp::evals()

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Eigenvalues of Hermitian *A*, as a real dynamic column vector

#### 6.1.3.63 hevects()

```
template<typename Derived >
cmat qpp::hevects (
            const Eigen::MatrixBase< Derived > & A )
```

Eigenvectors of Hermitian matrix.

**See also**

qpp::evects()

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Eigenvectors of Hermitian matrix *A*, as columns of a complex matrix

#### 6.1.3.64 inverse()

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::inverse (
            const Eigen::MatrixBase< Derived > & A )
```

Inverse.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Inverse of *A*, as a dynamic matrix over the same scalar field as *A*

#### 6.1.3.65 invperm()

```
std::vector<idx> qpp::invperm (
            const std::vector< idx > & perm ) [inline]
```

Inverse permutation.

**Parameters**

| | |
|---|---|
| *perm* | Permutation |

**Returns**

Inverse of the permutation *perm*

#### 6.1.3.66 ip() [1/2]

```
template<typename Derived >
dyn_col_vect<typename Derived::Scalar> qpp::ip (
            const Eigen::MatrixBase< Derived > & phi,
            const Eigen::MatrixBase< Derived > & psi,
            const std::vector< idx > & subsys,
            const std::vector< idx > & dims )
```

Generalized inner product.

**Parameters**

| | |
|---|---|
| *phi* | Column vector Eigen expression |
| *psi* | Column vector Eigen expression |
| *subsys* | Subsystem indexes over which *phi* is defined |
| *dims* | Dimensions of the multi-partite system |

**Returns**

Inner product $\langle \phi_{subsys}|\psi\rangle$, as a scalar or column vector over the remaining Hilbert space

#### 6.1.3.67 ip() [2/2]

```
template<typename Derived >
dyn_col_vect<typename Derived::Scalar> qpp::ip (
            const Eigen::MatrixBase< Derived > & phi,
            const Eigen::MatrixBase< Derived > & psi,
            const std::vector< idx > & subsys,
            idx d = 2 )
```

Generalized inner product.

**Parameters**

| | |
|---|---|
| *phi* | Column vector Eigen expression |
| *psi* | Column vector Eigen expression |
| *subsys* | Subsystem indexes over which *phi* is defined |
| *d* | Subsystem dimensions |

**Returns**

Inner product $\langle \phi_{subsys}|\psi\rangle$, as a scalar or column vector over the remaining Hilbert space

#### 6.1.3.68 isprime()

```
bool qpp::isprime (
            bigint p,
            idx k = 80 )  [inline]
```

Primality test based on the Miller-Rabin's algorithm.

**Parameters**

| | |
|---|---|
| *p* | Integer different from 0, 1 or -1 |
| *k* | Number of iterations. The probability of a false positive is $2^{-k}$. |

**Returns**

True if the number is (most-likely) prime, false otherwise

#### 6.1.3.69 kraus2choi()

```
cmat qpp::kraus2choi (
            const std::vector< cmat > & Ks )  [inline]
```

Choi matrix.

**See also**

qpp::choi2kraus()

Constructs the Choi matrix of the channel specified by the set of Kraus operators *Ks* in the standard operator basis $\{|i\rangle\langle j|\}$ ordered in lexicographical order, i.e. $|0\rangle\langle 0|$, $|0\rangle\langle 1|$ etc.

**Note**

The superoperator matrix $S$ and the Choi matrix $C$ are related by $S_{ab,mn} = C_{ma,nb}$

**Parameters**

| | |
|---|---|
| *Ks* | Set of Kraus operators |

**Returns**

Choi matrix

#### 6.1.3.70 kraus2super()

```
cmat qpp::kraus2super (
            const std::vector< cmat > & Ks )  [inline]
```

Superoperator matrix.

Constructs the superoperator matrix of the channel specified by the set of Kraus operators *Ks* in the standard operator basis $\{|i\rangle\langle j|\}$ ordered in lexicographical order, i.e. $|0\rangle\langle 0|$, $|0\rangle\langle 1|$ etc.

**Parameters**

| | |
|---|---|
| *Ks* | Set of Kraus operators |

**Returns**

Superoperator matrix

#### 6.1.3.71 kron() [1/4]

```
template<typename T >
dyn_mat<typename T::Scalar> qpp::kron (
            const T & head )
```

Kronecker product.

**See also**

qpp::kronpow()

Used to stop the recursion for the variadic template version of qpp::kron()

**Parameters**

| | |
|---|---|
| *head* | Eigen expression |

**Returns**

Its argument *head*

#### 6.1.3.72 kron() [2/4]

```
template<typename T , typename... Args>
dyn_mat<typename T::Scalar> qpp::kron (
            const T & head,
            const Args &... tail )
```

Kronecker product.

**See also**

qpp::kronpow()

**Parameters**

| | |
|---|---|
| *head* | Eigen expression |
| *tail* | Variadic Eigen expression (zero or more parameters) |

**Returns**

Kronecker product of all input parameters, evaluated from left to right, as a dynamic matrix over the same scalar field as its arguments

#### 6.1.3.73 kron() [3/4]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::kron (
            const std::vector< Derived > & As )
```

Kronecker product.

**See also**

qpp::kronpow()

**Parameters**

| | |
|---|---|
| *As* | std::vector of Eigen expressions |

**Returns**

Kronecker product of all elements in *As*, evaluated from left to right, as a dynamic matrix over the same scalar field as its arguments

#### 6.1.3.74 kron() [4/4]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::kron (
            const std::initializer_list< Derived > & As )
```

Kronecker product.

**See also**

qpp::kronpow()

**Parameters**

| | |
|---|---|
| *As* | std::initializer_list of Eigen expressions, such as *{A1*, A2, ... ,Ak} |

**Returns**

Kronecker product of all elements in *As*, evaluated from left to right, as a dynamic matrix over the same scalar field as its arguments

#### 6.1.3.75 kronpow()

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::kronpow (
```

```
            const Eigen::MatrixBase< Derived > & A,
            idx n )
```

Kronecker power.

**See also**

qpp::kron()

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *n* | Non-negative integer |

**Returns**

Kronecker product of *A* with itself *n* times $A^{\otimes n}$, as a dynamic matrix over the same scalar field as *A*

#### 6.1.3.76 lcm() [1/2]

```
bigint qpp::lcm (
            bigint a,
            bigint b ) [inline]
```

Least common multiple of two integers.

**See also**

qpp::gcd()

**Parameters**

| | |
|---|---|
| *a* | Integer |
| *b* | Integer |

**Returns**

Least common multiple of *a* and *b*

#### 6.1.3.77 lcm() [2/2]

```
bigint qpp::lcm (
            const std::vector< bigint > & as ) [inline]
```

Least common multiple of a list of integers.

**See also**

qpp::gcd()

**Parameters**

| | |
|---|---|
| *as* | List of integers |

**Returns**

Least common multiple of all numbers in *as*

#### 6.1.3.78 load()

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::load (
            const std::string & fname )
```

Loads Eigen matrix from a binary file (internal format) in double precision.

**See also**

qpp::save()

The template parameter cannot be automatically deduced and must be explicitly provided, depending on the scalar field of the matrix that is being loaded.

Example:

```
// loads a previously saved Eigen dynamic complex matrix from "input.bin"
cmat mat = load<cmat>("input.bin");
```

**Parameters**

| | |
|---|---|
| *fname* | Output file name |

#### 6.1.3.79 loadMATLAB() [1/2]

```
template<typename Derived >
std::enable_if<std::is_same<typename Derived::Scalar, cplx>::value, dyn_mat<cplx> >::type
qpp::loadMATLAB (
            const std::string & mat_file,
            const std::string & var_name )
```

Loads a complex Eigen dynamic matrix from a MATLAB .mat file,.

**See also**

qpp::saveMATLAB()

The template parameter cannot be automatically deduced and must be explicitly provided

Example:

```
// loads a previously saved Eigen ket
// from the MATLAB file "input.mat"
ket psi = loadMATLAB<ket>("input.mat");
```

**Template Parameters**

| | |
|---|---|
| *Derived* | Complex Eigen type |

**Parameters**

| | |
|---|---|
| *mat_file* | MATALB .mat file |
| *var_name* | Variable name in the .mat file representing the matrix to be loaded |

**Returns**

Eigen dynamic matrix

#### 6.1.3.80 loadMATLAB() [2/2]

```
template<typename Derived >
std::enable_if<!std::is_same<typename Derived::Scalar, cplx>::value, dyn_mat<typename Derived↩
::Scalar> >::type qpp::loadMATLAB (
            const std::string & mat_file,
            const std::string & var_name )
```

Loads a non-complex Eigen dynamic matrix from a MATLAB .mat file,.

**See also**

qpp::saveMATLAB()

The template parameter cannot be automatically deduced and must be explicitly provided

Example:

```
// loads a previously saved Eigen dynamic double matrix
// from the MATLAB file "input.mat"
dmat mat = loadMATLAB<dmat>("input.mat");
```

**Template Parameters**

| | |
|---|---|
| *Derived* | Non-complex Eigen type |

**Parameters**

| | |
|---|---|
| *mat_file* | MATALB .mat file |
| *var_name* | Variable name in the .mat file representing the matrix to be loaded |

**Returns**

Eigen dynamic matrix

#### 6.1.3.81 logdet()

```
template<typename Derived >
Derived::Scalar qpp::logdet (
            const Eigen::MatrixBase< Derived > & A )
```

Logarithm of the determinant.

Useful when the determinant overflows/underflows

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Logarithm of the determinant of *A*, as a scalar over the same scalar field as *A*

#### 6.1.3.82 logm()

```
template<typename Derived >
cmat qpp::logm (
            const Eigen::MatrixBase< Derived > & A )
```

Matrix logarithm.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Matrix logarithm of *A*

#### 6.1.3.83 lognegativity() [1/2]

```
template<typename Derived >
double qpp::lognegativity (
            const Eigen::MatrixBase< Derived > & A,
            const std::vector< idx > & dims )
```

Logarithmic negativity of the bi-partite mixed state *A*.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *dims* | Dimensions of the bi-partite system |

**Returns**

Logarithmic negativity, with the logarithm in base 2

#### 6.1.3.84 lognegativity() [2/2]

```
template<typename Derived >
double qpp::lognegativity (
            const Eigen::MatrixBase< Derived > & A,
            idx d = 2 )
```

Logarithmic negativity of the bi-partite mixed state *A*.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *d* | Subsystem dimensions |

**Returns**

Logarithmic negativity, with the logarithm in base 2

#### 6.1.3.85 marginalX()

```
std::vector<double> qpp::marginalX (
            const dmat & probXY )  [inline]
```

Marginal distribution.

**Parameters**

| | |
|---|---|
| *probXY* | Real matrix representing the joint probability distribution of *X* and *Y* in lexicographical order (*X* labels the rows, *Y* labels the columns) |

**Returns**

Real vector consisting of the marginal distribution of *X*

#### 6.1.3.86 marginalY()

```
std::vector<double> qpp::marginalY (
            const dmat & probXY )  [inline]
```

Marginal distribution.

**Parameters**

| | |
|---|---|
| *probXY* | Real matrix representing the joint probability distribution of *X* and *Y* in lexicographical order (*X* labels the rows, *Y* labels the columns) |

**Returns**

Real vector consisting of the marginal distribution of *Y*

#### 6.1.3.87 measure() [1/9]

```
template<typename Derived >
std::tuple<idx, std::vector<double>, std::vector<cmat> > qpp::measure (
            const Eigen::MatrixBase< Derived > & A,
            const std::vector< cmat > & Ks )
```

Measures the state vector or density operator *A* using the set of Kraus operators *Ks*.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *Ks* | Set of Kraus operators |

**Returns**

Tuple of: 1. Result of the measurement, 2. Vector of outcome probabilities, and 3. Vector of post-measurement normalized states

#### 6.1.3.88 measure() [2/9]

```
template<typename Derived >
std::tuple<idx, std::vector<double>, std::vector<cmat> > qpp::measure (
            const Eigen::MatrixBase< Derived > & A,
            const std::initializer_list< cmat > & Ks )
```

Measures the state vector or density matrix *A* using the set of Kraus operators *Ks*.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *Ks* | Set of Kraus operators |

**Returns**

Tuple of: 1. Result of the measurement, 2. Vector of outcome probabilities, and 3. Vector of post-measurement normalized states

#### 6.1.3.89 measure() [3/9]

```
template<typename Derived >
std::tuple<idx, std::vector<double>, std::vector<cmat> > qpp::measure (
            const Eigen::MatrixBase< Derived > & A,
            const cmat & U )
```

Measures the state vector or density matrix *A* in the orthonormal basis specified by the unitary matrix *U*.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *U* | Unitary matrix whose columns represent the measurement basis vectors |

**Returns**

Tuple of: 1. Result of the measurement, 2. Vector of outcome probabilities, and 3. Vector of post-measurement normalized states

#### 6.1.3.90 measure() [4/9]

```
template<typename Derived >
std::tuple<idx, std::vector<double>, std::vector<cmat> > qpp::measure (
            const Eigen::MatrixBase< Derived > & A,
            const std::vector< cmat > & Ks,
            const std::vector< idx > & target,
            const std::vector< idx > & dims )
```

Measures the part *subsys* of the multi-partite state vector or density matrix *A* using the set of Kraus operators *Ks*.

**See also**

qpp::measure_seq()

**Note**

The dimension of all *Ks* must match the dimension of *target*. The measurement is destructive, i.e. the measured subsystems are traced away.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *Ks* | Set of Kraus operators |
| *target* | Subsystem indexes that are measured |
| *dims* | Dimensions of the multi-partite system |

**Returns**

Tuple of: 1. Result of the measurement, 2. Vector of outcome probabilities, and 3. Vector of post-measurement normalized states

#### 6.1.3.91 measure() [5/9]

```
template<typename Derived >
std::tuple<idx, std::vector<double>, std::vector<cmat> > qpp::measure (
            const Eigen::MatrixBase< Derived > & A,
            const std::initializer_list< cmat > & Ks,
            const std::vector< idx > & target,
            const std::vector< idx > & dims )
```

Measures the part *target* of the multi-partite state vector or density matrix *A* using the set of Kraus operators *Ks*.

**See also**

qpp::measure_seq()

**Note**

The dimension of all *Ks* must match the dimension of *target*. The measurement is destructive, i.e. the measured subsystems are traced away.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *Ks* | Set of Kraus operators |
| *target* | Subsystem indexes that are measured |
| *dims* | Dimensions of the multi-partite system |

**Returns**

Tuple of: 1. Result of the measurement, 2. Vector of outcome probabilities, and 3. Vector of post-measurement normalized states

#### 6.1.3.92 measure() [6/9]

```
template<typename Derived >
std::tuple<idx, std::vector<double>, std::vector<cmat> > qpp::measure (
            const Eigen::MatrixBase< Derived > & A,
            const std::vector< cmat > & Ks,
            const std::vector< idx > & target,
            idx d = 2 )
```

Measures the part *target* of the multi-partite state vector or density matrix *A* using the set of Kraus operators *Ks*.

**See also**

qpp::measure_seq()

**Note**

The dimension of all *Ks* must match the dimension of *target*. The measurement is destructive, i.e. the measured subsystems are traced away.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *Ks* | Set of Kraus operators |
| *target* | Subsystem indexes that are measured |
| *d* | Subsystem dimensions |

**Returns**

Tuple of: 1. Result of the measurement, 2. Vector of outcome probabilities, and 3. Vector of post-measurement normalized states

#### 6.1.3.93 measure() [7/9]

```
template<typename Derived >
std::tuple<idx, std::vector<double>, std::vector<cmat> > qpp::measure (
            const Eigen::MatrixBase< Derived > & A,
            const std::initializer_list< cmat > & Ks,
            const std::vector< idx > & target,
            idx d = 2 )
```

Measures the part *target* of the multi-partite state vector or density matrix *A* using the set of Kraus operators *Ks*.

**See also**

qpp::measure_seq()

**Note**

The dimension of all *Ks* must match the dimension of *target*. The measurement is destructive, i.e. the measured subsystems are traced away.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *Ks* | Set of Kraus operators |
| *target* | Subsystem indexes that are measured |
| *d* | Subsystem dimensions |

**Returns**

Tuple of: 1. Result of the measurement, 2. Vector of outcome probabilities, and 3. Vector of post-measurement normalized states

#### 6.1.3.94 measure() [8/9]

```
template<typename Derived >
std::tuple<idx, std::vector<double>, std::vector<cmat> > qpp::measure (
            const Eigen::MatrixBase< Derived > & A,
            const cmat & V,
            const std::vector< idx > & target,
            const std::vector< idx > & dims )
```

Measures the part *target* of the multi-partite state vector or density matrix *A* in the orthonormal basis or rank-1 projectors specified by the columns of the matrix *V*.

**See also**

qpp::measure_seq()

**Note**

The dimension of *V* must match the dimension of *target*. The measurement is destructive, i.e. the measured subsystems are traced away.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *V* | Matrix whose columns represent the measurement basis vectors or the bra parts of the rank-1 projectors |
| *target* | Subsystem indexes that are measured |
| *dims* | Dimensions of the multi-partite system |

**Returns**

Tuple of: 1. Result of the measurement, 2. Vector of outcome probabilities, and 3. Vector of post-measurement normalized states

#### 6.1.3.95 measure() [9/9]

```
template<typename Derived >
std::tuple<idx, std::vector<double>, std::vector<cmat> > qpp::measure (
            const Eigen::MatrixBase< Derived > & A,
            const cmat & V,
            const std::vector< idx > & target,
            idx d = 2 )
```

Measures the part *target* of the multi-partite state vector or density matrix *A* in the orthonormal basis or rank-1 projectors specified by the columns of the matrix *V*.

**See also**

qpp::measure_seq()

**Note**

The dimension of *V* must match the dimension of *target*. The measurement is destructive, i.e. the measured subsystems are traced away.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *V* | Matrix whose columns represent the measurement basis vectors or the bra parts of the rank-1 projectors |
| *target* | Subsystem indexes that are measured |
| *d* | Subsystem dimensions |

**Returns**

Tuple of: 1. Result of the measurement, 2. Vector of outcome probabilities, and 3. Vector of post-measurement normalized states

#### 6.1.3.96 measure_seq() [1/2]

```
template<typename Derived >
std::tuple<std::vector<idx>, double, cmat> qpp::measure_seq (
            const Eigen::MatrixBase< Derived > & A,
            std::vector< idx > target,
            std::vector< idx > dims )
```

Sequentially measures the part *target* of the multi-partite state vector or density matrix *A* in the computational basis.

**See also**

qpp::measure()

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *target* | Subsystem indexes that are measured |
| *dims* | Dimensions of the multi-partite system |

**Returns**

Tuple of: 1. Vector of outcome results of the measurement (ordered in increasing order with respect to *target*, i.e. first measurement result corresponds to the subsystem with the smallest index), 2. Outcome probability, and 3. Post-measurement normalized state

#### 6.1.3.97 measure_seq() [2/2]

```
template<typename Derived >
std::tuple<std::vector<idx>, double, cmat> qpp::measure_seq (
            const Eigen::MatrixBase< Derived > & A,
            std::vector< idx > target,
            idx d = 2 )
```

Sequentially measures the part *target* of the multi-partite state vector or density matrix *A* in the computational basis.

**See also**

qpp::measure()

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *target* | Subsystem indexes that are measured |
| *d* | Subsystem dimensions |

**Returns**

Tuple of: 1. Vector of outcome results of the measurement (ordered in increasing order with respect to *target*, i.e. first measurement result corresponds to the subsystem with the smallest index), 2. Outcome probability, and 3. Post-measurement normalized state

#### 6.1.3.98 mket() [1/2]

```
ket qpp::mket (
            const std::vector< idx > & mask,
            const std::vector< idx > & dims ) [inline]
```

Multi-partite qudit ket.

**See also**

qpp::operator "" _ket()

Constructs the multi-partite qudit ket $|\mathrm{mask}\rangle$, where *mask* is a std::vector of non-negative integers. Each element in *mask* has to be smaller than the corresponding element in *dims*.

**Parameters**

| | |
|---|---|
| *mask* | std::vector of non-negative integers |
| *dims* | Dimensions of the multi-partite system |

**Returns**

Multi-partite qudit state vector, as a complex dynamic column vector

#### 6.1.3.99 mket() [2/2]

```
ket qpp::mket (
            const std::vector< idx > & mask,
            idx d = 2 )  [inline]
```

Multi-partite qudit ket.

**See also**

qpp::operator "" _ket()

Constructs the multi-partite qudit ket $|\mathrm{mask}\rangle$, all subsystem having equal dimension *d*. *mask* is a std::vector of non-negative integers, and each element in *mask* has to be strictly smaller than *d*.

**Parameters**

| | |
|---|---|
| *mask* | std::vector of non-negative integers |
| *d* | Subsystem dimensions |

**Returns**

Multi-partite qudit state vector, as a complex dynamic column vector

#### 6.1.3.100 modinv()

```
bigint qpp::modinv (
            bigint a,
            bigint p )  [inline]
```

Modular inverse of *a* mod *p*.

**See also**

qpp::egcd()

**Note**

*a* and *p* must be co-prime

**Parameters**

| | |
|---|---|
| *a* | Non-negative integer |
| *p* | Non-negative integer |

**Returns**

Modular inverse $a^{-1} \mod p$

#### 6.1.3.101 modmul()

```
bigint qpp::modmul (
            bigint a,
            bigint b,
            bigint p )  [inline]
```

Modular multiplication without overflow.

Computes $ab \bmod p$ without overflow

**Parameters**

| | |
|---|---|
| *a* | Integer |
| *b* | Integer |
| *p* | Positive integer |

**Returns**

$ab \bmod p$ avoiding overflow

#### 6.1.3.102 modpow()

```
bigint qpp::modpow (
            bigint a,
```

```
            bigint n,
            bigint p )  [inline]
```

Fast integer power modulo *p* based on the SQUARE-AND-MULTIPLY algorithm.

**Note**

Uses qpp::modmul() that avoids overflows

Computes $a^n \bmod p$

**Parameters**

| | |
|---|---|
| *a* | Non-negative integer |
| *n* | Non-negative integer |
| *p* | Strictly positive integer |

**Returns**

$a^n \bmod p$

#### 6.1.3.103 mprj() [1/2]

```
cmat qpp::mprj (
            const std::vector< idx > & mask,
            const std::vector< idx > & dims )  [inline]
```

Projector onto multi-partite qudit ket.

**See also**

qpp::operator "" _prj()

Constructs the projector onto the multi-partite qudit ket $|\mathrm{mask}\rangle$, where *mask* is a std::vector of non-negative integers. Each element in *mask* has to be smaller than the corresponding element in *dims*.

**Parameters**

| | |
|---|---|
| *mask* | std::vector of non-negative integers |
| *dims* | Dimensions of the multi-partite system |

**Returns**

Projector onto multi-partite qudit state vector, as a complex dynamic matrix

#### 6.1.3.104 mprj() [2/2]

```
cmat qpp::mprj (
            const std::vector< idx > & mask,
            idx d = 2 ) [inline]
```

Projector onto multi-partite qudit ket.

**See also**

qpp::operator "" _prj()

Constructs the projector onto the multi-partite qudit ket $|\mathrm{mask}\rangle$, all subsystem having equal dimension *d*. *mask* is a std::vector of non-negative integers, and each element in *mask* has to be strictly smaller than *d*.

**Parameters**

| | |
|---|---|
| *mask* | std::vector of non-negative integers |
| *d* | Subsystem dimensions |

**Returns**

Projector onto multi-partite qudit state vector, as a complex dynamic matrix

#### 6.1.3.105 multiidx2n()

```
idx qpp::multiidx2n (
            const std::vector< idx > & midx,
            const std::vector< idx > & dims ) [inline]
```

Multi-index to non-negative integer index.

**See also**

qpp::n2multiidx()

Uses standard lexicographical order, i.e. 00...0, 00...1 etc.

**Parameters**

| | |
|---|---|
| *midx* | Multi-index |
| *dims* | Dimensions of the multi-partite system |

**Returns**

Non-negative integer index

#### 6.1.3.106 n2multiidx()

```
std::vector<idx> qpp::n2multiidx (
            idx n,
            const std::vector< idx > & dims )  [inline]
```

Non-negative integer index to multi-index.

**See also**

qpp::multiidx2n()

Uses standard lexicographical order, i.e. 00...0, 00...1 etc.

**Parameters**

| | |
|---|---|
| *n* | Non-negative integer index |
| *dims* | Dimensions of the multi-partite system |

**Returns**

Multi-index of the same size as *dims*

#### 6.1.3.107 negativity() [1/2]

```
template<typename Derived >
double qpp::negativity (
            const Eigen::MatrixBase< Derived > & A,
            const std::vector< idx > & dims )
```

Negativity of the bi-partite mixed state *A*.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *dims* | Dimensions of the bi-partite system |

**Returns**

Negativity

#### 6.1.3.108 negativity() [2/2]

```
template<typename Derived >
double qpp::negativity (
```

```
            const Eigen::MatrixBase< Derived > & A,
            idx d = 2 )
```

Negativity of the bi-partite mixed state *A*.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *d* | Subsystem dimensions |

**Returns**

Negativity

#### 6.1.3.109 norm()

```
template<typename Derived >
double qpp::norm (
            const Eigen::MatrixBase< Derived > & A )
```

Frobenius norm.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Frobenius norm of *A*

#### 6.1.3.110 normalize()

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::normalize (
            const Eigen::MatrixBase< Derived > & A )
```

Normalizes state vector (column or row vector) or density matrix.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Normalized state vector or density matrix

#### 6.1.3.111 omega()

```
cplx qpp::omega (
            idx D )  [inline]
```

D-th root of unity.

**Parameters**

| | |
|---|---|
| *D* | Non-negative integer |

**Returns**

D-th root of unity $\exp(2\pi i/D)$

#### 6.1.3.112 operator"""" _i()

```
constexpr cplx qpp::operator"" _i (
            long double x )  [inline], [noexcept]
```

User-defined literal for complex $i = \sqrt{-1}$ (real overload)

Example:

```
cplx z = 4.5_i; // type of z is std::complex<double>
```

#### 6.1.3.113 powm()

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::powm (
            const Eigen::MatrixBase< Derived > & A,
            idx n )
```

Fast matrix power based on the SQUARE-AND-MULTIPLY algorithm.

**See also**

qpp::spectralpowm()

Explicitly multiplies the matrix *A* with itself *n* times. By convention $A^0 = I$.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *n* | Non-negative integer |

**Returns**

Matrix power $A^n$, as a dynamic matrix over the same scalar field as *A*

#### 6.1.3.114 prj()

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::prj (
            const Eigen::MatrixBase< Derived > & A )
```

Projector.

Normalized projector onto state vector

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Projector onto the state vector *A*, or the matrix *Zero* if *A* has norm zero, as a dynamic matrix over the same scalar field as *A*

#### 6.1.3.115 prod() [1/3]

```
template<typename Derived >
Derived::Scalar qpp::prod (
            const Eigen::MatrixBase< Derived > & A )
```

Element-wise product of *A*.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Element-wise product of *A*, as a scalar over the same scalar field as *A*

#### 6.1.3.116 prod() [2/3]

```
template<typename InputIterator >
std::iterator_traits<InputIterator>::value_type qpp::prod (
            InputIterator first,
            InputIterator last )
```

Element-wise product of an STL-like range.

**Parameters**

| | |
|---|---|
| *first* | Iterator to the first element of the range |
| *last* | Iterator to the last element of the range |

**Returns**

Element-wise product of the range, as a scalar over the same scalar field as the range

#### 6.1.3.117 prod() [3/3]

```
template<typename Container >
Container::value_type qpp::prod (
            const Container & c,
            typename std::enable_if< is_iterable< Container >::value >::type * = nullptr )
```

Element-wise product of the elements of an STL-like container.

**Parameters**

| | |
|---|---|
| *c* | STL-like container |

**Returns**

Element-wise product of the elements of the container, as a scalar over the same scalar field as the container

#### 6.1.3.118 ptrace() [1/2]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::ptrace (
            const Eigen::MatrixBase< Derived > & A,
            const std::vector< idx > & target,
            const std::vector< idx > & dims )
```

Partial trace.

**See also**

qpp::ptrace1(), qpp::ptrace2()

Partial trace of the multi-partite state vector or density matrix over the list *target* of subsystems

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *target* | Subsystem indexes |
| *dims* | Dimensions of the multi-partite system |

**Returns**

Partial trace $Tr_{subsys}(\cdot)$ over the subsytems *target* in a multi-partite system, as a dynamic matrix over the same scalar field as *A*

#### 6.1.3.119 ptrace() [2/2]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::ptrace (
            const Eigen::MatrixBase< Derived > & A,
            const std::vector< idx > & target,
            idx d = 2 )
```

Partial trace.

**See also**

qpp::ptrace1(), qpp::ptrace2()

Partial trace of the multi-partite state vector or density matrix over the list *target* of subsystems

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *target* | Subsystem indexes |
| *d* | Subsystem dimensions |

**Returns**

Partial trace $Tr_{subsys}(\cdot)$ over the subsytems *target* in a multi-partite system, as a dynamic matrix over the same scalar field as *A*

#### 6.1.3.120 ptrace1() [1/2]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::ptrace1 (
            const Eigen::MatrixBase< Derived > & A,
            const std::vector< idx > & dims )
```

Partial trace.

**See also**

qpp::ptrace2()

Partial trace over the first subsystem of bi-partite state vector or density matrix

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *dims* | Dimensions of the bi-partite system |

**Returns**

Partial trace $Tr_A(\cdot)$ over the first subsytem $A$ in a bi-partite system $A \otimes B$, as a dynamic matrix over the same scalar field as *A*

#### 6.1.3.121 ptrace1() [2/2]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::ptrace1 (
            const Eigen::MatrixBase< Derived > & A,
            idx d = 2 )
```

Partial trace.

**See also**

qpp::ptrace2()

Partial trace over the first subsystem of bi-partite state vector or density matrix

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *d* | Subsystem dimensions |

**Returns**

Partial trace $Tr_A(\cdot)$ over the first subsytem $A$ in a bi-partite system $A \otimes B$, as a dynamic matrix over the same scalar field as *A*

#### 6.1.3.122 ptrace2() [1/2]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::ptrace2 (
            const Eigen::MatrixBase< Derived > & A,
            const std::vector< idx > & dims )
```

Partial trace.

**See also**

qpp::ptrace1()

Partial trace over the second subsystem of bi-partite state vector or density matrix

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *dims* | Dimensions of the bi-partite system |

**Returns**

Partial trace $Tr_B(\cdot)$ over the second subsytem $B$ in a bi-partite system $A \otimes B$, as a dynamic matrix over the same scalar field as *A*

#### 6.1.3.123 ptrace2() [2/2]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::ptrace2 (
            const Eigen::MatrixBase< Derived > & A,
            idx d = 2 )
```

Partial trace.

**See also**

qpp::ptrace1()

Partial trace over the second subsystem of bi-partite state vector or density matrix

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *d* | Subsystem dimensions |

**Returns**

Partial trace $Tr_B(\cdot)$ over the second subsytem $B$ in a bi-partite system $A \otimes B$, as a dynamic matrix over the same scalar field as *A*

#### 6.1.3.124 ptranspose() [1/2]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::ptranspose (
```

```
            const Eigen::MatrixBase< Derived > & A,
            const std::vector< idx > & target,
            const std::vector< idx > & dims )
```

Partial transpose.

Partial transpose of the multi-partite state vector or density matrix over the list *target* of subsystems

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *target* | Subsystem indexes |
| *dims* | Dimensions of the multi-partite system |

**Returns**

Partial transpose $(\cdot)^{T_{subsys}}$ over the subsytems *target* in a multi-partite system, as a dynamic matrix over the same scalar field as *A*

#### 6.1.3.125 ptranspose() [2/2]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::ptranspose (
            const Eigen::MatrixBase< Derived > & A,
            const std::vector< idx > & target,
            idx d = 2 )
```

Partial transpose.

Partial transpose of the multi-partite state vector or density matrix over the list *target* of subsystems

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *target* | Subsystem indexes |
| *d* | Subsystem dimensions |

**Returns**

Partial transpose $(\cdot)^{T_{subsys}}$ over the subsytems *target* in a multi-partite system, as a dynamic matrix over the same scalar field as *A*

#### 6.1.3.126 QFT()

```
template<typename Derived >
dyn_col_vect<typename Derived::Scalar> qpp::QFT (
```

```
const Eigen::MatrixBase< Derived > & A,
idx d = 2,
bool swap = true )
```

Qudit quantum Fourier transform.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *d* | Subsystem dimensions |
| *swap* | Swaps the qubits/qudits at the end (true by default) |

**Returns**

Qudit quantum Fourier transform applied on *A*

#### 6.1.3.127 qmutualinfo() [1/2]

```
template<typename Derived >
double qpp::qmutualinfo (
            const Eigen::MatrixBase< Derived > & A,
            const std::vector< idx > & subsysA,
            const std::vector< idx > & subsysB,
            const std::vector< idx > & dims )
```

Quantum mutual information between 2 subsystems of a composite system.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *subsysA* | Indexes of the first subsystem |
| *subsysB* | Indexes of the second subsystem |
| *dims* | Dimensions of the multi-partite system |

**Returns**

Mutual information between the 2 subsystems

#### 6.1.3.128 qmutualinfo() [2/2]

```
template<typename Derived >
double qpp::qmutualinfo (
            const Eigen::MatrixBase< Derived > & A,
            const std::vector< idx > & subsysA,
            const std::vector< idx > & subsysB,
            idx d = 2 )
```

Quantum mutual information between 2 subsystems of a composite system.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *subsysA* | Indexes of the first subsystem |
| *subsysB* | Indexes of the second subsystem |
| *d* | Subsystem dimensions |

**Returns**

Mutual information between the 2 subsystems

#### 6.1.3.129 rand() [1/5]

```
double qpp::rand (
            double a,
            double b )  [inline]
```

Generates a random real number uniformly distributed in the interval [a, b)

**Parameters**

| | |
|---|---|
| *a* | Beginning of the interval, belongs to it |
| *b* | End of the interval, does not belong to it |

**Returns**

Random real number (double) uniformly distributed in the interval [a, b)

#### 6.1.3.130 rand() [2/5]

```
bigint qpp::rand (
            bigint a,
            bigint b )  [inline]
```

Generates a random big integer uniformly distributed in the interval [a, b].

**Note**

To avoid ambiguity with double qpp::rand(double, double) cast at least one of the arguments to qpp::bigint

**Parameters**

| | |
|---|---|
| *a* | Beginning of the interval, belongs to it |
| *b* | End of the interval, belongs to it |

**Returns**

Random big integer uniformly distributed in the interval [a, b]

#### 6.1.3.131 rand() [3/5]

```
template<typename Derived >
Derived qpp::rand (
            idx rows QPP_UNUSED_,
            idx cols QPP_UNUSED_,
            double a QPP_UNUSED_ = 0,
            double b QPP_UNUSED_ = 1 )
```

Generates a random matrix with entries uniformly distributed in the interval [a, b)

If complex, then both real and imaginary parts are uniformly distributed in [a, b)

This is the generic version that always throws qpp::Exception::Type::UNDEFINED_TYPE. It is specialized only for qpp::dmat and qpp::cmat

#### 6.1.3.132 rand() [4/5]

```
template<>
dmat qpp::rand (
            idx rows ,
            idx cols ,
            double a ,
            double b ) [inline]
```

Generates a random real matrix with entries uniformly distributed in the interval [a, b), specialization for double matrices (qpp::dmat)

The template parameter cannot be automatically deduced and must be explicitly provided

Example:

```
// generates a 3 x 3 random Eigen::MatrixXd,
// with entries uniformly distributed in [-1,1)
dmat mat = rand<dmat>(3, 3, -1, 1);
```

**Parameters**

| | |
|---|---|
| *rows* | Number of rows of the random generated matrix |
| *cols* | Number of columns of the random generated matrix |
| *a* | Beginning of the interval, belongs to it |
| *b* | End of the interval, does not belong to it |

**Returns**

Random real matrix

#### 6.1.3.133 rand() [5/5]

```
template<>
cmat qpp::rand (
            idx rows ,
            idx cols ,
            double a ,
            double b )  [inline]
```

Generates a random complex matrix with entries (both real and imaginary) uniformly distributed in the interval [a, b), specialization for complex matrices (qpp::cmat)

The template parameter cannot be automatically deduced and must be explicitly provided

Example:

```
// generates a 3 x 3 random Eigen::MatrixXcd,
// with entries (both real and imaginary) uniformly distributed in [-1,1)
cmat mat = rand<cmat>(3, 3, -1, 1);
```

**Parameters**

| | |
|---|---|
| *rows* | Number of rows of the random generated matrix |
| *cols* | Number of columns of the random generated matrix |
| *a* | Beginning of the interval, belongs to it |
| *b* | End of the interval, does not belong to it |

**Returns**

Random complex matrix

#### 6.1.3.134 randH()

```
cmat qpp::randH (
            idx D = 2 )  [inline]
```

Generates a random Hermitian matrix.

**Parameters**

| | |
|---|---|
| *D* | Dimension of the Hilbert space |

**Returns**

Random Hermitian matrix

#### 6.1.3.135 randidx()

```
idx qpp::randidx (
            idx a = std::numeric_limits<idx>::min(),
            idx b = std::numeric_limits<idx>::max() )  [inline]
```

Generates a random index (idx) uniformly distributed in the interval [a, b].

**Parameters**

| | |
|---|---|
| *a* | Beginning of the interval, belongs to it |
| *b* | End of the interval, belongs to it |

**Returns**

Random index (idx) uniformly distributed in the interval [a, b]

#### 6.1.3.136 randket()

```
ket qpp::randket (
            idx D = 2 )  [inline]
```

Generates a random normalized ket (pure state vector)

**Parameters**

| | |
|---|---|
| *D* | Dimension of the Hilbert space |

**Returns**

Random normalized ket

#### 6.1.3.137 randkraus()

```
std::vector<cmat> qpp::randkraus (
            idx N,
            idx D = 2 )  [inline]
```

Generates a set of random Kraus operators.

**Note**

The set of Kraus operators satisfy the closure condition $\sum_i K_i^\dagger K_i = I$

**Parameters**

| | |
|---|---|
| *N* | Number of Kraus operators |
| *D* | Dimension of the Hilbert space |

**Returns**

Set of *N* Kraus operators satisfying the closure condition

#### 6.1.3.138 randn() [1/4]

```
template<typename Derived >
Derived qpp::randn (
            idx rows QPP_UNUSED_,
            idx cols QPP_UNUSED_,
            double mean QPP_UNUSED_ = 0,
            double sigma QPP_UNUSED_ = 1 )
```

Generates a random matrix with entries normally distributed in N(mean, sigma)

If complex, then both real and imaginary parts are normally distributed in N(mean, sigma)

This is the generic version that always throws qpp::Exception::Type::UNDEFINED_TYPE. It is specialized only for qpp::dmat and qpp::cmat

#### 6.1.3.139 randn() [2/4]

```
template<>
dmat qpp::randn (
            idx rows ,
            idx cols ,
            double mean ,
            double sigma )  [inline]
```

Generates a random real matrix with entries normally distributed in N(mean, sigma), specialization for double matrices (qpp::dmat)

The template parameter cannot be automatically deduced and must be explicitly provided

Example:

```
// generates a 3 x 3 random Eigen::MatrixXd,
// with entries normally distributed in N(0,2)
dmat mat = randn<dmat>(3, 3, 0, 2);
```

**Parameters**

| | |
|---|---|
| *rows* | Number of rows of the random generated matrix |
| *cols* | Number of columns of the random generated matrix |
| *mean* | Mean |
| *sigma* | Standard deviation |

**Returns**

Random real matrix

#### 6.1.3.140 randn() [3/4]

```
template<>
cmat qpp::randn (
            idx rows ,
            idx cols ,
            double mean ,
            double sigma )  [inline]
```

Generates a random complex matrix with entries (both real and imaginary) normally distributed in N(mean, sigma), specialization for complex matrices (qpp::cmat)

The template parameter cannot be automatically deduced and must be explicitly provided

Example:

```
// generates a 3 x 3 random Eigen::MatrixXcd,
// with entries (both real and imaginary) normally distributed in N(0,2)
cmat mat = randn<cmat>(3, 3, 0, 2);
```

**Parameters**

| | |
|---|---|
| *rows* | Number of rows of the random generated matrix |
| *cols* | Number of columns of the random generated matrix |
| *mean* | Mean |
| *sigma* | Standard deviation |

**Returns**

Random complex matrix

#### 6.1.3.141 randn() [4/4]

```
double qpp::randn (
            double mean = 0,
            double sigma = 1 )  [inline]
```

Generates a random real number (double) normally distributed in N(mean, sigma)

**Parameters**

| | |
|---|---|
| *mean* | Mean |
| *sigma* | Standard deviation |

**Returns**

Random real number normally distributed in N(mean, sigma)

#### 6.1.3.142 randperm()

```
std::vector<idx> qpp::randperm (
            idx N ) [inline]
```

Generates a random uniformly distributed permutation.

Uses Knuth shuffle method (as implemented by std::shuffle), so that all permutations are equally probable

**Parameters**

| | |
|---|---|
| *N* | Size of the permutation |

**Returns**

Random permutation of size *N*

#### 6.1.3.143 randprime()

```
bigint qpp::randprime (
            bigint a,
            bigint b,
            idx N = 1000 ) [inline]
```

Generates a random big prime uniformly distributed in the interval [a, b].

**Parameters**

| | |
|---|---|
| *a* | Beginning of the interval, belongs to it |
| *b* | End of the interval, belongs to it |
| *N* | Maximum number of candidates |

**Returns**

Random big integer uniformly distributed in the interval [a, b]

#### 6.1.3.144 randprob()

```
std::vector<double> qpp::randprob (
            idx N ) [inline]
```

Generates a random probability vector uniformly distributed over the probability simplex.

**Parameters**

| | |
|---|---|
| *N* | Size of the probability vector |

**Returns**

Random probability vector

#### 6.1.3.145 randrho()

```
cmat qpp::randrho (
            idx D = 2 ) [inline]
```

Generates a random density matrix.

**Parameters**

| | |
|---|---|
| *D* | Dimension of the Hilbert space |

**Returns**

Random density matrix

#### 6.1.3.146 randU()

```
cmat qpp::randU (
            idx D = 2 ) [inline]
```

Generates a random unitary matrix.

**Parameters**

| | |
|---|---|
| *D* | Dimension of the Hilbert space |

**Returns**

Random unitary

#### 6.1.3.147 randV()

```
cmat qpp::randV (
            idx Din,
            idx Dout )  [inline]
```

Generates a random isometry matrix.

**Parameters**

| | |
|---|---|
| *Din* | Size of the input Hilbert space |
| *Dout* | Size of the output Hilbert space |

**Returns**

Random isometry matrix

#### 6.1.3.148 renyi() [1/2]

```
template<typename Derived >
double qpp::renyi (
            const Eigen::MatrixBase< Derived > & A,
            double alpha )
```

Renyi- $\alpha$ entropy of the density matrix *A*, for $\alpha \geq 0$.

**Note**

When $\alpha \to 1$ the Renyi entropy converges to the von-Neumann entropy, with the logarithm in base 2

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *alpha* | Non-negative real number, use qpp::infty for $\alpha = \infty$ |

**Returns**

Renyi- $\alpha$ entropy, with the logarithm in base 2

#### 6.1.3.149 renyi() [2/2]

```
double qpp::renyi (
            const std::vector< double > & prob,
            double alpha ) [inline]
```

Renyi- $\alpha$ entropy of the probability distribution *prob*, for $\alpha \geq 0$.

**Note**

When $\alpha \to 1$ the Renyi entropy converges to the Shannon entropy, with the logarithm in base 2

**Parameters**

| | |
|---|---|
| *prob* | Real probability vector |
| *alpha* | Non-negative real number, use qpp::infty for $\alpha = \infty$ |

**Returns**

Renyi- $\alpha$ entropy, with the logarithm in base 2

#### 6.1.3.150 reshape()

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::reshape (
            const Eigen::MatrixBase< Derived > & A,
            idx rows,
            idx cols )
```

Reshape.

Uses column-major order when reshaping (same as MATLAB)

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *rows* | Number of rows of the reshaped matrix |
| *cols* | Number of columns of the reshaped matrix |

**Returns**

Reshaped matrix with *rows* rows and *cols* columns, as a dynamic matrix over the same scalar field as *A*

#### 6.1.3.151 rho2bloch()

```
template<typename Derived >
std::vector<double> qpp::rho2bloch (
            const Eigen::MatrixBase< Derived > & A )
```

Computes the 3-dimensional real Bloch vector corresponding to the qubit density matrix *A*.

**See also**

qpp::bloch2rho()

**Note**

It is implicitly assumed that the density matrix is Hermitian

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

3-dimensional Bloch vector

#### 6.1.3.152 rho2pure()

```
template<typename Derived >
dyn_col_vect<typename Derived::Scalar> qpp::rho2pure (
            const Eigen::MatrixBase< Derived > & A )
```

Finds the pure state representation of a matrix proportional to a projector onto a pure state.

**Note**

No purity check is done, the input state *A* must have rank one, otherwise the function returns the first non-zero eigenvector of *A*

**Parameters**

| | |
|---|---|
| *A* | Eigen expression, assumed to be proportional to a projector onto a pure state, i.e. *A* is assumed to have rank one |

**Returns**

The unique non-zero eigenvector of *A* (up to a phase), as a dynamic column vector over the same scalar field as *A*

#### 6.1.3.153 save()

```
template<typename Derived >
void qpp::save (
            const Eigen::MatrixBase< Derived > & A,
            const std::string & fname )
```

Saves Eigen expression to a binary file (internal format) in double precision.

**See also**

qpp::load()

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *fname* | Output file name |

#### 6.1.3.154 saveMATLAB() [1/2]

```
template<typename Derived >
std::enable_if< std::is_same<typename Derived::Scalar, cplx>::value>::type qpp::saveMATLAB (
            const Eigen::MatrixBase< Derived > & A,
            const std::string & mat_file,
            const std::string & var_name,
            const std::string & mode )
```

Saves a complex Eigen dynamic matrix to a MATLAB .mat file,.

**See also**

qpp::loadMATLAB()

**Template Parameters**

| | |
|---|---|
| *Complex* | Eigen type |

**Parameters**

| | |
|---|---|
| *A* | Eigen expression over the complex field |

**Parameters**

| | |
|---|---|
| *mat_file* | MATALB .mat file |
| *var_name* | Variable name in the .mat file representing the matrix to be saved |
| *mode* | Saving mode (append, overwrite etc.), see MATLAB *matOpen()* documentation for details |

#### 6.1.3.155 saveMATLAB() [2/2]

```
template<typename Derived >
std::enable_if< !std::is_same<typename Derived::Scalar, cplx>::value>::type qpp::saveMATLAB (
            const Eigen::MatrixBase< Derived > & A,
            const std::string & mat_file,
            const std::string & var_name,
            const std::string & mode )
```

Saves a non-complex Eigen dynamic matrix to a MATLAB .mat file,.

**See also**

qpp::loadMATLAB()

**Template Parameters**

| | |
|---|---|
| *Npn-complex* | Eigen type |

**Parameters**

| | |
|---|---|
| *A* | Non-complex Eigen expression |
| *mat_file* | MATALB .mat file |
| *var_name* | Variable name in the .mat file representing the matrix to be saved |
| *mode* | Saving mode (append, overwrite etc.), see MATLAB *matOpen()* documentation for details |

#### 6.1.3.156 schatten()

```
template<typename Derived >
double qpp::schatten (
            const Eigen::MatrixBase< Derived > & A,
            double p )
```

Schatten matrix norm.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *p* | Real number, greater or equal to 1, use qpp::infty for $p = \infty$ |

**Returns**

Schatten-*p* matrix norm of *A*

#### 6.1.3.157 schmidtA() [1/2]

```
template<typename Derived >
cmat qpp::schmidtA (
            const Eigen::MatrixBase< Derived > & A,
            const std::vector< idx > & dims )
```

Schmidt basis on Alice side.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *dims* | Dimensions of the bi-partite system |

**Returns**

Unitary matrix $U$ whose columns represent the Schmidt basis vectors on Alice side.

#### 6.1.3.158 schmidtA() [2/2]

```
template<typename Derived >
cmat qpp::schmidtA (
            const Eigen::MatrixBase< Derived > & A,
            idx d = 2 )
```

Schmidt basis on Alice side.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *d* | Subsystem dimensions |

**Returns**

Unitary matrix $U$ whose columns represent the Schmidt basis vectors on Alice side.

#### 6.1.3.159 schmidtB() [1/2]

```
template<typename Derived >
cmat qpp::schmidtB (
```

```
            const Eigen::MatrixBase< Derived > & A,
            const std::vector< idx > & dims )
```

Schmidt basis on Bob side.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *dims* | Dimensions of the bi-partite system |

**Returns**

Unitary matrix $V$ whose columns represent the Schmidt basis vectors on Bob side.

#### 6.1.3.160 schmidtB() [2/2]

```
template<typename Derived >
cmat qpp::schmidtB (
            const Eigen::MatrixBase< Derived > & A,
            idx d = 2 )
```

Schmidt basis on Bob side.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *d* | Subsystem dimensions |

**Returns**

Unitary matrix $V$ whose columns represent the Schmidt basis vectors on Bob side.

#### 6.1.3.161 schmidtcoeffs() [1/2]

```
template<typename Derived >
dyn_col_vect<double> qpp::schmidtcoeffs (
            const Eigen::MatrixBase< Derived > & A,
            const std::vector< idx > & dims )
```

Schmidt coefficients of the bi-partite pure state *A*.

**Note**

The sum of the squares of the Schmidt coefficients equals 1

**See also**

qpp::schmidtprobs()

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *dims* | Dimensions of the bi-partite system |

**Returns**

Schmidt coefficients of *A*, ordered in decreasing order, as a real dynamic column vector

#### 6.1.3.162 schmidtcoeffs() [2/2]

```
template<typename Derived >
dyn_col_vect<double> qpp::schmidtcoeffs (
            const Eigen::MatrixBase< Derived > & A,
            idx d = 2 )
```

Schmidt coefficients of the bi-partite pure state *A*.

**Note**

The sum of the squares of the Schmidt coefficients equals 1

**See also**

qpp::schmidtprobs()

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *d* | Subsystem dimensions |

**Returns**

Schmidt coefficients of *A*, ordered in decreasing order, as a real dynamic column vector

#### 6.1.3.163 schmidtprobs() [1/2]

```
template<typename Derived >
std::vector<double> qpp::schmidtprobs (
            const Eigen::MatrixBase< Derived > & A,
            const std::vector< idx > & dims )
```

Schmidt probabilities of the bi-partite pure state *A*.

Defined as the squares of the Schmidt coefficients. The sum of the Schmidt probabilities equals 1.

**See also**

qpp::schmidtcoeffs()

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *dims* | Dimensions of the bi-partite system |

**Returns**

Real vector consisting of the Schmidt probabilites of *A*, ordered in decreasing order

#### 6.1.3.164 schmidtprobs() [2/2]

```
template<typename Derived >
std::vector<double> qpp::schmidtprobs (
            const Eigen::MatrixBase< Derived > & A,
            idx d = 2 )
```

Schmidt probabilities of the bi-partite pure state *A*.

Defined as the squares of the Schmidt coefficients. The sum of the Schmidt probabilities equals 1.

**See also**

qpp::schmidtcoeffs()

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *d* | Subsystem dimensions |

**Returns**

Real vector consisting of the Schmidt probabilites of *A*, ordered in decreasing order

#### 6.1.3.165 sigma()

```
template<typename Container >
double qpp::sigma (
            const std::vector< double > & prob,
            const Container & X,
            typename std::enable_if< is_iterable< Container >::value >::type * = nullptr )
```

Standard deviation.

**Parameters**

| | |
|---|---|
| *prob* | Real probability vector representing the probability distribution of *X* |
| *X* | Real random variable values represented by an STL-like container |

**Returns**

Standard deviation of *X*

#### 6.1.3.166 sinm()

```
template<typename Derived >
cmat qpp::sinm (
            const Eigen::MatrixBase< Derived > & A )
```

Matrix sin.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Matrix sine of *A*

#### 6.1.3.167 spectralpowm()

```
template<typename Derived >
cmat qpp::spectralpowm (
            const Eigen::MatrixBase< Derived > & A,
            const cplx z )
```

Matrix power.

**See also**

qpp::powm()

Uses the spectral decomposition of *A* to compute the matrix power. By convention $A^0 = I$.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *z* | Complex number |

**Returns**

Matrix power $A^z$

#### 6.1.3.168 sqrtm()

```
template<typename Derived >
cmat qpp::sqrtm (
            const Eigen::MatrixBase< Derived > & A )
```

Matrix square root.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Matrix square root of *A*

#### 6.1.3.169 sum() [1/3]

```
template<typename Derived >
Derived::Scalar qpp::sum (
            const Eigen::MatrixBase< Derived > & A )
```

Element-wise sum of *A*.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Element-wise sum of *A*, as a scalar over the same scalar field as *A*

#### 6.1.3.170 sum() [2/3]

```
template<typename InputIterator >
std::iterator_traits<InputIterator>::value_type qpp::sum (
            InputIterator first,
            InputIterator last )
```

Element-wise sum of an STL-like range.

**Parameters**

| | |
|---|---|
| *first* | Iterator to the first element of the range |
| *last* | Iterator to the last element of the range |

**Returns**

Element-wise sum of the range, as a scalar over the same scalar field as the range

#### 6.1.3.171 sum() [3/3]

```
template<typename Container >
Container::value_type qpp::sum (
            const Container & c,
            typename std::enable_if< is_iterable< Container >::value >::type *  = nullptr )
```

Element-wise sum of the elements of an STL-like container.

**Parameters**

| | |
|---|---|
| *c* | STL-like container |

**Returns**

Element-wise sum of the elements of the container, as a scalar over the same scalar field as the container

#### 6.1.3.172 super2choi()

```
cmat qpp::super2choi (
            const cmat & A )  [inline]
```

Converts superoperator matrix to Choi matrix.

**See also**

qpp::choi2super()

**Parameters**

| | |
|---|---|
| *A* | Superoperator matrix |

**Returns**

Choi matrix

#### 6.1.3.173 svals()

```
template<typename Derived >
dyn_col_vect<double> qpp::svals (
            const Eigen::MatrixBase< Derived > & A )
```

Singular values.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Singular values of *A*, ordered in decreasing order, as a real dynamic column vector

#### 6.1.3.174 svd()

```
template<typename Derived >
std::tuple<cmat, dyn_col_vect<double>, cmat> qpp::svd (
            const Eigen::MatrixBase< Derived > & A )
```

Full singular value decomposition.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Tuple of: 1. Left sigular vectors of *A*, as columns of a complex dynamic matrix, 2. Singular values of *A*, ordered in decreasing order, as a real dynamic column vector, and 3. Right singular vectors of *A*, as columns of a complex dynamic matrix

#### 6.1.3.175 svdU()

```
template<typename Derived >
cmat qpp::svdU (
            const Eigen::MatrixBase< Derived > & A )
```

Left singular vectors.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Complex dynamic matrix, whose columns are the left singular vectors of *A*

#### 6.1.3.176 svdV()

```
template<typename Derived >
cmat qpp::svdV (
            const Eigen::MatrixBase< Derived > & A )
```

Right singular vectors.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Complex dynamic matrix, whose columns are the right singular vectors of *A*

#### 6.1.3.177 syspermute() [1/2]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::syspermute (
            const Eigen::MatrixBase< Derived > & A,
            const std::vector< idx > & perm,
            const std::vector< idx > & dims )
```

Subsystem permutation.

Permutes the subsystems of a state vector or density matrix. The qubit *perm*[*i*] is permuted to the location *i*.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *perm* | Permutation |
| *dims* | Dimensions of the multi-partite system |

#### Returns

Permuted system, as a dynamic matrix over the same scalar field as *A*

### 6.1.3.178 syspermute() [2/2]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::syspermute (
            const Eigen::MatrixBase< Derived > & A,
            const std::vector< idx > & perm,
            idx d = 2 )
```

Subsystem permutation.

Permutes the subsystems of a state vector or density matrix. The qubit *perm*[*i*] is permuted to the location *i*.

#### Parameters

| | |
|---|---|
| *A* | Eigen expression |
| *perm* | Permutation |
| *d* | Subsystem dimensions |

#### Returns

Permuted system, as a dynamic matrix over the same scalar field as *A*

### 6.1.3.179 TFQ()

```
template<typename Derived >
dyn_col_vect<typename Derived::Scalar> qpp::TFQ (
            const Eigen::MatrixBase< Derived > & A,
            idx d = 2,
            bool swap = true )
```

Inverse (adjoint) qudit quantum Fourier transform.

#### Parameters

| | |
|---|---|
| *A* | Eigen expression |
| *d* | Subsystem dimensions |
| *swap* | Swaps the qubits/qudits at the end (true by default) |

#### Returns

Inverse (adjoint) qudit quantum Fourier transform applied on *A*

#### 6.1.3.180 trace()

```
template<typename Derived >
Derived::Scalar qpp::trace (
            const Eigen::MatrixBase< Derived > & A )
```

Trace.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Trace of *A*, as a scalar over the same scalar field as *A*

#### 6.1.3.181 transpose()

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::transpose (
            const Eigen::MatrixBase< Derived > & A )
```

Transpose.

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |

**Returns**

Transpose of *A*, as a dynamic matrix over the same scalar field as *A*

#### 6.1.3.182 tsallis() [1/2]

```
template<typename Derived >
double qpp::tsallis (
            const Eigen::MatrixBase< Derived > & A,
            double q )
```

Tsallis- $q$ entropy of the density matrix *A*, for $q \geq 0$.

**Note**

When $q \rightarrow 1$ the Tsallis entropy converges to the von-Neumann entropy, with the logarithm in base $e$

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *q* | Non-negative real number |

**Returns**

Tsallis- $q$ entropy

#### 6.1.3.183 tsallis() [2/2]

```
double qpp::tsallis (
            const std::vector< double > & prob,
            double q ) [inline]
```

Tsallis- $q$ entropy of the probability distribution *prob*, for $q \geq 0$.

**Note**

When $q \to 1$ the Tsallis entropy converges to the Shannon entropy, with the logarithm in base $e$

**Parameters**

| | |
|---|---|
| *prob* | Real probability vector |
| *q* | Non-negative real number |

**Returns**

Tsallis- $q$ entropy

#### 6.1.3.184 uniform()

```
std::vector<double> qpp::uniform (
            idx N ) [inline]
```

Uniform probability distribution vector.

**Parameters**

| | |
|---|---|
| *N* | Size of the alphabet |

**Returns**

Real vector consisting of a uniform distribution of size *N*

#### 6.1.3.185 var()

```
template<typename Container >
double qpp::var (
            const std::vector< double > & prob,
            const Container & X,
            typename std::enable_if< is_iterable< Container >::value >::type * = nullptr )
```

Variance.

**Parameters**

| | |
|---|---|
| *prob* | Real probability vector representing the probability distribution of *X* |
| *X* | Real random variable values represented by an STL-like container |

**Returns**

Variance of *X*

#### 6.1.3.186 x2contfrac()

```
std::vector<int> qpp::x2contfrac (
            double x,
            idx N,
            idx cut = 1e5 ) [inline]
```

Simple continued fraction expansion.

**See also**

qpp::contfrac2x()

**Parameters**

| | |
|---|---|
| *x* | Real number |
| *N* | Maximum number of terms in the expansion |
| *cut* | Stop the expansion when the next term is greater than *cut* |

**Returns**

Integer vector containing the simple continued fraction expansion of *x*. If there are *M* less than *N* terms in the expansion, a shorter vector with *M* components is returned.

### 6.1.4 Variable Documentation

#### 6.1.4.1 chop

```
constexpr double qpp::chop = 1e-10
```

Used in qpp::disp() for setting to zero numbers that have their absolute value smaller than qpp::chop.

#### 6.1.4.2 ee

```
constexpr double qpp::ee = 2.718281828459045235360287471352662497
```

Base of natural logarithm, $e$.

#### 6.1.4.3 infty

```
constexpr double qpp::infty = std::numeric_limits<double>::max()
```

Used to denote infinity in double precision.

#### 6.1.4.4 maxn

```
constexpr idx qpp::maxn = 64
```

Maximum number of allowed qubits/qudits (subsystems)

Used internally to allocate arrays on the stack (for performance reasons):

#### 6.1.4.5 pi

```
constexpr double qpp::pi = 3.141592653589793238462643383279502884
```

$\pi$

## 6.2 qpp::exception Namespace Reference

Quantum++ exception hierarchy namespace.

## Classes

- class CustomException
  *Custom exception.*
- class DimsInvalid
  *Invalid dimension(s) exception.*
- class DimsMismatchCvector
  *Dimension(s) mismatch column vector size exception.*
- class DimsMismatchMatrix
  *Dimension(s) mismatch matrix size exception.*
- class DimsMismatchRvector
  *Dimension(s) mismatch row vector size exception.*
- class DimsMismatchVector
  *Dimension(s) mismatch vector size exception.*
- class DimsNotEqual
  *Dimensions not equal exception.*
- class Duplicates
  *System (e.g. std::vector) has duplicates exception.*
- class Exception
  *Base class for generating Quantum++ custom exceptions.*
- class InvalidIterator
  *Invalid iterator.*
- class MatrixMismatchSubsys
  *Matrix mismatch subsystems exception.*
- class MatrixNotCvector
  *Matrix is not a column vector exception.*
- class MatrixNotRvector
  *Matrix is not a row vector exception.*
- class MatrixNotSquare
  *Matrix is not square exception.*
- class MatrixNotSquareNorCvector
  *Matrix is not square nor column vector exception.*
- class MatrixNotSquareNorRvector
  *Matrix is not square nor row vector exception.*
- class MatrixNotSquareNorVector
  *Matrix is not square nor vector exception.*
- class MatrixNotVector
  *Matrix is not a vector exception.*
- class NoCodeword
  *Codeword does not exist exception.*
- class NotBipartite
  *Not bi-partite exception.*
- class NotImplemented
  *Code not yet implemented.*
- class NotQubitCvector
  *Column vector is not 2 x 1 exception.*
- class NotQubitMatrix
  *Matrix is not 2 x 2 exception.*
- class NotQubitRvector
  *Row vector is not 1 x 2 exception.*
- class NotQubitSubsys

*Subsystems are not qubits exception.*
- class NotQubitVector

  *Vector is not 2 x 1 nor 1 x 2 exception.*
- class OutOfRange

  *Argument out of range exception.*
- class PermInvalid

  *Invalid permutation exception.*
- class PermMismatchDims

  *Permutation mismatch dimensions exception.*
- class QuditAlreadyMeasured

  *Qudit was already measured exception.*
- class SizeMismatch

  *Size mismatch exception.*
- class SubsysMismatchDims

  *Subsystems mismatch dimensions exception.*
- class TypeMismatch

  *Type mismatch exception.*
- class UndefinedType

  *Not defined for this type exception.*
- class Unknown

  *Unknown exception.*
- class ZeroSize

  *Object has zero size exception.*

### 6.2.1 Detailed Description

Quantum++ exception hierarchy namespace.

## 6.3 qpp::experimental Namespace Reference

Experimental/test functions/classes, do not use or modify.

### 6.3.1 Detailed Description

Experimental/test functions/classes, do not use or modify.

## 6.4 qpp::internal Namespace Reference

Internal utility functions, do not use them directly or modify them.

## Classes

- struct Display_Impl_
- class EqualEigen

  *Functor for comparing Eigen expressions for equality.*
- class HashEigen

  *Functor for hashing Eigen expressions.*
- class IOManipEigen
- class IOManipPointer
- class IOManipRange
- class Singleton

  *Singleton policy class, used internally to implement the singleton pattern via CRTP (Curiously recurring template pattern)*

## Functions

- template< class T >
  void hash_combine (std::size_t &seed, const T &v)
- void n2multiidx (idx n, idx numdims, const idx ∗const dims, idx ∗result) noexcept
- idx multiidx2n (const idx ∗const midx, idx numdims, const idx ∗const dims) noexcept
- template< typename Derived >
  bool check_square_mat (const Eigen::MatrixBase< Derived > &A)
- template< typename Derived >
  bool check_vector (const Eigen::MatrixBase< Derived > &A)
- template< typename Derived >
  bool check_rvector (const Eigen::MatrixBase< Derived > &A)
- template< typename Derived >
  bool check_cvector (const Eigen::MatrixBase< Derived > &A)
- template< typename T >
  bool check_nonzero_size (const T &x) noexcept
- template< typename T1 , typename T2 >
  bool check_matching_sizes (const T1 &lhs, const T2 &rhs) noexcept
- bool check_dims (const std::vector< idx > &dims)
- template< typename Derived >
  bool check_dims_match_mat (const std::vector< idx > &dims, const Eigen::MatrixBase< Derived > &A)
- template< typename Derived >
  bool check_dims_match_cvect (const std::vector< idx > &dims, const Eigen::MatrixBase< Derived > &A)
- template< typename Derived >
  bool check_dims_match_rvect (const std::vector< idx > &dims, const Eigen::MatrixBase< Derived > &A)
- bool check_eq_dims (const std::vector< idx > &dims, idx dim) noexcept
- bool check_no_duplicates (std::vector< idx > v)
- bool check_subsys_match_dims (const std::vector< idx > &subsys, const std::vector< idx > &dims)
- template< typename Derived >
  bool check_qubit_matrix (const Eigen::MatrixBase< Derived > &A) noexcept
- template< typename Derived >
  bool check_qubit_cvector (const Eigen::MatrixBase< Derived > &A) noexcept
- template< typename Derived >
  bool check_qubit_rvector (const Eigen::MatrixBase< Derived > &A) noexcept
- template< typename Derived >
  bool check_qubit_vector (const Eigen::MatrixBase< Derived > &A) noexcept
- bool check_perm (const std::vector< idx > &perm)
- template< typename Derived1 , typename Derived2 >
  dyn_mat< typename Derived1::Scalar > kron2 (const Eigen::MatrixBase< Derived1 > &A, const Eigen::↩
  MatrixBase< Derived2 > &B)

- template<typename Derived1 , typename Derived2 >
  dyn_mat< typename Derived1::Scalar > dirsum2 (const Eigen::MatrixBase< Derived1 > &A, const Eigen↩ ::MatrixBase< Derived2 > &B)
- template<typename T >
  void variadic_vector_emplace (std::vector< T > &)
- template<typename T , typename First , typename... Args>
  void variadic_vector_emplace (std::vector< T > &v, First &&first, Args &&... args)
- idx get_num_subsys (idx D, idx d)
- idx get_dim_subsys (idx sz, idx N)

### 6.4.1 Detailed Description

Internal utility functions, do not use them directly or modify them.

### 6.4.2 Function Documentation

#### 6.4.2.1 check_cvector()

```
template<typename Derived >
bool qpp::internal::check_cvector (
            const Eigen::MatrixBase< Derived > & A )
```

#### 6.4.2.2 check_dims()

```
bool qpp::internal::check_dims (
            const std::vector< idx > & dims )  [inline]
```

#### 6.4.2.3 check_dims_match_cvect()

```
template<typename Derived >
bool qpp::internal::check_dims_match_cvect (
            const std::vector< idx > & dims,
            const Eigen::MatrixBase< Derived > & A )
```

#### 6.4.2.4 check_dims_match_mat()

```
template<typename Derived >
bool qpp::internal::check_dims_match_mat (
            const std::vector< idx > & dims,
            const Eigen::MatrixBase< Derived > & A )
```

#### 6.4.2.5 check_dims_match_rvect()

```
template<typename Derived >
bool qpp::internal::check_dims_match_rvect (
            const std::vector< idx > & dims,
            const Eigen::MatrixBase< Derived > & A )
```

#### 6.4.2.6 check_eq_dims()

```
bool qpp::internal::check_eq_dims (
            const std::vector< idx > & dims,
            idx dim )  [inline], [noexcept]
```

#### 6.4.2.7 check_matching_sizes()

```
template<typename T1 , typename T2 >
bool qpp::internal::check_matching_sizes (
            const T1 & lhs,
            const T2 & rhs )  [noexcept]
```

#### 6.4.2.8 check_no_duplicates()

```
bool qpp::internal::check_no_duplicates (
            std::vector< idx > v )  [inline]
```

#### 6.4.2.9 check_nonzero_size()

```
template<typename T >
bool qpp::internal::check_nonzero_size (
            const T & x )  [noexcept]
```

#### 6.4.2.10 check_perm()

```
bool qpp::internal::check_perm (
            const std::vector< idx > & perm )  [inline]
```

#### 6.4.2.11 check_qubit_cvector()

```
template<typename Derived >
bool qpp::internal::check_qubit_cvector (
            const Eigen::MatrixBase< Derived > & A ) [noexcept]
```

#### 6.4.2.12 check_qubit_matrix()

```
template<typename Derived >
bool qpp::internal::check_qubit_matrix (
            const Eigen::MatrixBase< Derived > & A ) [noexcept]
```

#### 6.4.2.13 check_qubit_rvector()

```
template<typename Derived >
bool qpp::internal::check_qubit_rvector (
            const Eigen::MatrixBase< Derived > & A ) [noexcept]
```

#### 6.4.2.14 check_qubit_vector()

```
template<typename Derived >
bool qpp::internal::check_qubit_vector (
            const Eigen::MatrixBase< Derived > & A ) [noexcept]
```

#### 6.4.2.15 check_rvector()

```
template<typename Derived >
bool qpp::internal::check_rvector (
            const Eigen::MatrixBase< Derived > & A )
```

#### 6.4.2.16 check_square_mat()

```
template<typename Derived >
bool qpp::internal::check_square_mat (
            const Eigen::MatrixBase< Derived > & A )
```

#### 6.4.2.17 check_subsys_match_dims()

```
bool qpp::internal::check_subsys_match_dims (
            const std::vector< idx > & subsys,
            const std::vector< idx > & dims ) [inline]
```

#### 6.4.2.18 check_vector()

```
template<typename Derived >
bool qpp::internal::check_vector (
            const Eigen::MatrixBase< Derived > & A )
```

#### 6.4.2.19 dirsum2()

```
template<typename Derived1 , typename Derived2 >
dyn_mat<typename Derived1::Scalar> qpp::internal::dirsum2 (
            const Eigen::MatrixBase< Derived1 > & A,
            const Eigen::MatrixBase< Derived2 > & B )
```

#### 6.4.2.20 get_dim_subsys()

```
idx qpp::internal::get_dim_subsys (
            idx sz,
            idx N ) [inline]
```

#### 6.4.2.21 get_num_subsys()

```
idx qpp::internal::get_num_subsys (
            idx D,
            idx d ) [inline]
```

#### 6.4.2.22 hash_combine()

```
template<class T >
void qpp::internal::hash_combine (
            std::size_t & seed,
            const T & v )
```

#### 6.4.2.23 kron2()

```
template<typename Derived1 , typename Derived2 >
dyn_mat<typename Derived1::Scalar> qpp::internal::kron2 (
            const Eigen::MatrixBase< Derived1 > & A,
            const Eigen::MatrixBase< Derived2 > & B )
```

#### 6.4.2.24 multiidx2n()

```
idx qpp::internal::multiidx2n (
            const idx *const midx,
            idx numdims,
            const idx *const dims )  [inline], [noexcept]
```

#### 6.4.2.25 n2multiidx()

```
void qpp::internal::n2multiidx (
            idx n,
            idx numdims,
            const idx *const dims,
            idx * result )  [inline], [noexcept]
```

#### 6.4.2.26 variadic_vector_emplace() [1/2]

```
template<typename T >
void qpp::internal::variadic_vector_emplace (
            std::vector< T > &  )
```

#### 6.4.2.27 variadic_vector_emplace() [2/2]

```
template<typename T , typename First , typename... Args>
void qpp::internal::variadic_vector_emplace (
            std::vector< T > & v,
            First && first,
            Args &&... args )
```

## 6.5 qpp::literals Namespace Reference

### Functions

- constexpr cplx operator"" _i (unsigned long long int x) noexcept

  *User-defined literal for complex $i = \sqrt{-1}$ (integer overload)*
- template<char... Bits>
  ket operator"" _ket ()

  *Multi-partite qubit ket user-defined literal.*
- template<char... Bits>
  bra operator"" _bra ()

  *Multi-partite qubit bra user-defined literal.*
- template<char... Bits>
  cmat operator"" _prj ()

  *Multi-partite qubit projector user-defined literal.*

### 6.5.1 Function Documentation

#### 6.5.1.1 operator"""" _bra()

```
template<char... Bits>
bra qpp::literals::operator"" _bra ( )
```

Multi-partite qubit bra user-defined literal.

**See also**

qpp::mket() and qpp::adjoint()

Constructs the multi-partite qubit bra $\langle \mathrm{Bits}|$

**Template Parameters**

| | |
|---|---|
| *Bits* | String of binary numbers representing the qubit bra |

**Returns**

Multi-partite qubit bra, as a complex dynamic row vector

#### 6.5.1.2 operator"""" _i()

```
constexpr cplx qpp::literals::operator"" _i (
            unsigned long long int x )  [inline], [noexcept]
```

User-defined literal for complex $i = \sqrt{-1}$ (integer overload)

Example:

```
cplx z = 4_i; // type of z is std::complex<double>
```

#### 6.5.1.3 operator"""" _ket()

```
template<char... Bits>
ket qpp::literals::operator"" _ket ( )
```

Multi-partite qubit ket user-defined literal.

**See also**

qpp::mket()

Constructs the multi-partite qubit ket $|\mathrm{Bits}\rangle$

**Template Parameters**

| | |
|---|---|
| *Bits* | String of binary numbers representing the qubit ket |

**Returns**

Multi-partite qubit ket, as a complex dynamic column vector

#### 6.5.1.4 operator"""" _prj()

```
template<char... Bits>
cmat qpp::literals::operator"" _prj ( )
```

Multi-partite qubit projector user-defined literal.

**See also**

qpp::mprj()

Constructs the multi-partite qubit projector $|\mathrm{Bits}\rangle\langle\mathrm{Bits}|$ (in the computational basis)

**Template Parameters**

| | |
|---|---|
| *Bits* | String of binary numbers representing the qubit state to project on |

**Returns**

Multi-partite qubit projector, as a complex dynamic matrix

# Chapter 7

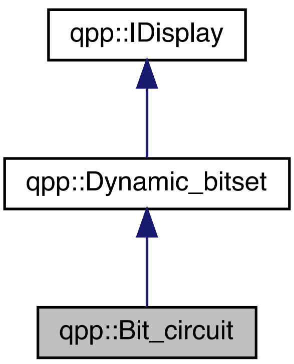
# Class Documentation

## 7.1 qpp::Bit_circuit Class Reference

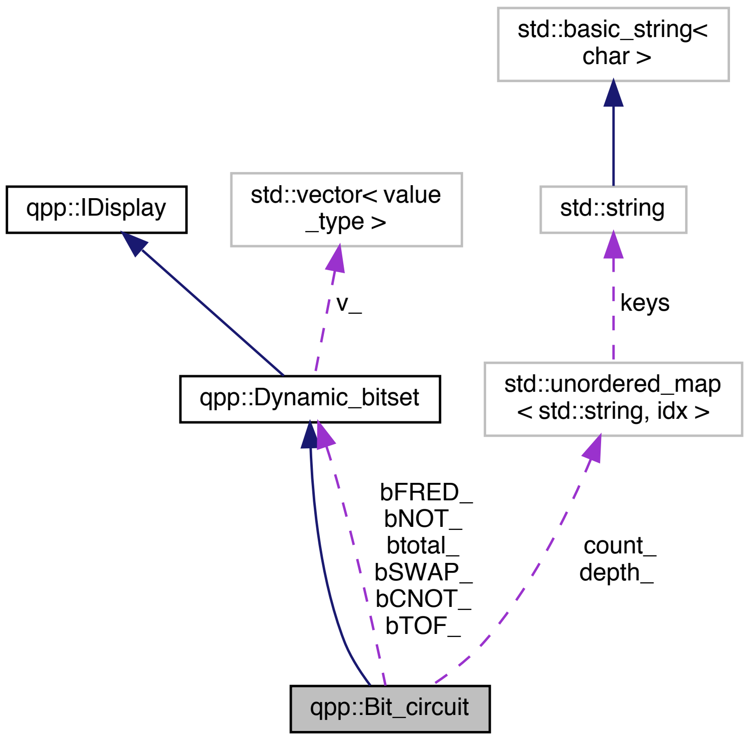
Classical reversible circuit simulator.

```
#include <classes/reversible.h>
```

Inheritance diagram for qpp::Bit_circuit:



Collaboration diagram for qpp::Bit_circuit:




## Public Member Functions

- Bit_circuit (idx n)

  *Constructs a bit circuit instance.*
- Bit_circuit (const Dynamic_bitset &dynamic_bitset)

  *Conversion constructor, used to initialize a qpp::Bit_circuit with a qpp::Dynamic_bitset.*
- Bit_circuit & X (idx i)

  *Bit flip.*
- virtual ∼Bit_circuit ()=default

  *Default virtual destructor.*
- Bit_circuit & NOT (idx i)

  *Bit flip.*
- Bit_circuit & CNOT (idx ctrl, idx target)

  *Controlled-NOT.*
- Bit_circuit & TOF (idx i, idx j, idx k)

  *Toffoli gate.*
- Bit_circuit & SWAP (idx i, idx j)

  *Swap bits.*
- Bit_circuit & FRED (idx i, idx j, idx k)

  *Fredkin gate (Controlled-SWAP)*
- Bit_circuit & reset () noexcept

Reset the circuit all zero, clear all gates.

- idx get_gate_count (const std::string &name={}) const

  *Bit circuit gate count.*
- idx get_gate_depth (const std::string &name={}) const

  *Bit circuit gate depth.*

## Private Attributes

- std::unordered_map< std::string, idx > count_ {}

  *gate counts*
- std::unordered_map< std::string, idx > depth_ {}

  *gate depths*
- Dynamic_bitset bNOT_
- Dynamic_bitset bCNOT_
- Dynamic_bitset bSWAP_
- Dynamic_bitset bTOF_
- Dynamic_bitset bFRED_
- Dynamic_bitset btotal_

  *used for depth calculations*

## Additional Inherited Members

### 7.1.1 Detailed Description

Classical reversible circuit simulator.

### 7.1.2 Constructor & Destructor Documentation

#### 7.1.2.1 Bit_circuit() [1/2]

```
qpp::Bit_circuit::Bit_circuit (
            idx n )  [inline], [explicit]
```

Constructs a bit circuit instance.

**Parameters**

| | |
|---|---|
| *n* | Number of classical bits |

#### 7.1.2.2 Bit_circuit() [2/2]

```
qpp::Bit_circuit::Bit_circuit (
            const Dynamic_bitset & dynamic_bitset )  [inline], [explicit]
```

Conversion constructor, used to initialize a qpp::Bit_circuit with a qpp::Dynamic_bitset.

**Parameters**

| | |
|---|---|
| *dynamic_bitset* | Dynamic bitset |

#### 7.1.2.3 ∼Bit_circuit()

```
virtual qpp::Bit_circuit::∼Bit_circuit ( ) [virtual], [default]
```

Default virtual destructor.

### 7.1.3 Member Function Documentation

#### 7.1.3.1 CNOT()

```
Bit_circuit& qpp::Bit_circuit::CNOT (
            idx ctrl,
            idx target ) [inline]
```

Controlled-NOT.

**Parameters**

| | |
|---|---|
| *ctrl* | Control bit index |
| *target* | Target bit index |

**Returns**

Reference to the current instance

#### 7.1.3.2 FRED()

```
Bit_circuit& qpp::Bit_circuit::FRED (
            idx i,
            idx j,
            idx k ) [inline]
```

Fredkin gate (Controlled-SWAP)

**Parameters**

| | |
|---|---|
| *i* | Control bit index |
| *j* | Target first bit index |
| *k* | Target second bit index |

**Returns**

Reference to the current instance

#### 7.1.3.3 get_gate_count()

```
idx qpp::Bit_circuit::get_gate_count (
            const std::string & name = {} ) const  [inline]
```

Bit circuit gate count.

**Note**

If *name* is empty (default), returns the total gate count of the circuit

**Parameters**

| | |
|---|---|
| *name* | Gate name (optional). Possible names are NOT (X), CNOT, SWAP, TOF, FRED. |

**Returns**

Gate count

#### 7.1.3.4 get_gate_depth()

```
idx qpp::Bit_circuit::get_gate_depth (
            const std::string & name = {} ) const  [inline]
```

Bit circuit gate depth.

**Note**

If *name* is empty (default), returns the total gate depth of the circuit

**Parameters**

| | |
|---|---|
| *name* | Gate name (optional). Possible names are NOT (X), CNOT, SWAP, TOF, FRED. |

**Returns**

Gate depth

#### 7.1.3.5 NOT()

```
Bit_circuit& qpp::Bit_circuit::NOT (
             idx i )  [inline]
```

Bit flip.

**See also**

qpp::Bit_circuit::X()

**Parameters**

| | |
|---|---|
| *i* | Bit position in the circuit |

**Returns**

Reference to the current instance

#### 7.1.3.6 reset()

```
Bit_circuit& qpp::Bit_circuit::reset ( )  [inline], [noexcept]
```

Reset the circuit all zero, clear all gates.

**Returns**

Reference to the current instance

#### 7.1.3.7 SWAP()

```
Bit_circuit& qpp::Bit_circuit::SWAP (
             idx i,
             idx j )  [inline]
```

Swap bits.

**Parameters**

| | |
|---|---|
| *i* | Bit index |
| *j* | Bit index |

**Returns**

Reference to the current instance

#### 7.1.3.8 TOF()

```
Bit_circuit& qpp::Bit_circuit::TOF (
            idx i,
            idx j,
            idx k )  [inline]
```

Toffoli gate.

**Parameters**

| | |
|---|---|
| *i* | Control first bit index |
| *j* | Control second bit index |
| *k* | Target bit index |

**Returns**

Reference to the current instance

#### 7.1.3.9 X()

```
Bit_circuit& qpp::Bit_circuit::X (
            idx i )  [inline]
```

Bit flip.

**See also**

qpp::Bit_circuit::NOT()

**Parameters**

| | |
|---|---|
| *i* | Bit position in the circuit |

**Returns**

Reference to the current instance

### 7.1.4 Member Data Documentation

#### 7.1.4.1 bCNOT_

`Dynamic_bitset qpp::Bit_circuit::bCNOT_ [private]`

#### 7.1.4.2 bFRED_

`Dynamic_bitset qpp::Bit_circuit::bFRED_ [private]`

#### 7.1.4.3 bNOT_

`Dynamic_bitset qpp::Bit_circuit::bNOT_ [private]`

#### 7.1.4.4 bSWAP_

`Dynamic_bitset qpp::Bit_circuit::bSWAP_ [private]`

#### 7.1.4.5 bTOF_

`Dynamic_bitset qpp::Bit_circuit::bTOF_ [private]`

#### 7.1.4.6 btotal_

`Dynamic_bitset qpp::Bit_circuit::btotal_ [private]`

used for depth calculations

#### 7.1.4.7 count_

```
std::unordered_map<std::string, idx> qpp::Bit_circuit::count_ {} [private]
```

gate counts

#### 7.1.4.8 depth_

```
std::unordered_map<std::string, idx> qpp::Bit_circuit::depth_ {} [private]
```

gate depths

The documentation for this class was generated from the following file:

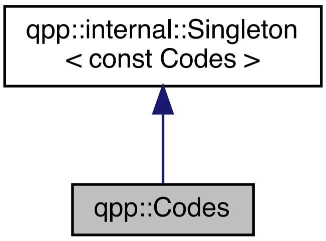
- classes/reversible.h

## 7.2 qpp::Codes Class Reference

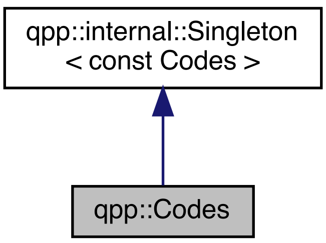
const Singleton class that defines quantum error correcting codes

```
#include <classes/codes.h>
```

Inheritance diagram for qpp::Codes:




Collaboration diagram for qpp::Codes:



### Public Types

- enum Type { Type::FIVE_QUBIT = 1, Type::SEVEN_QUBIT_STEANE, Type::NINE_QUBIT_SHOR }
  *Code types, add more codes here if needed.*

### Public Member Functions

- ket codeword (Type type, idx i) const
  *Returns the codeword of the specified code type.*

### Private Member Functions

- Codes ()
  *Default constructor.*
- ∼Codes ()=default
  *Default destructor.*

### Friends

- class internal::Singleton< const Codes >

### Additional Inherited Members

## 7.2.1 Detailed Description

const Singleton class that defines quantum error correcting codes

## 7.2.2 Member Enumeration Documentation

### 7.2.2.1 Type

```
enum qpp::Codes::Type [strong]
```

Code types, add more codes here if needed.

**See also**

qpp::Codes::codeword()

**Enumerator**

| | |
|---|---|
| FIVE_QUBIT | [[5,1,3]] qubit code |
| SEVEN_QUBIT_STEANE | [[7,1,3]] Steane qubit code |
| NINE_QUBIT_SHOR | [[9,1,3]] Shor qubit code |

### 7.2.3 Constructor & Destructor Documentation

#### 7.2.3.1 Codes()

```
qpp::Codes::Codes ( )  [inline], [private]
```

Default constructor.

#### 7.2.3.2 ∼Codes()

```
qpp::Codes::∼Codes ( )  [private], [default]
```

Default destructor.

### 7.2.4 Member Function Documentation

#### 7.2.4.1 codeword()

```
ket qpp::Codes::codeword (
             Type type,
             idx i ) const  [inline]
```

Returns the codeword of the specified code type.

**See also**

qpp::Codes::Type

**Parameters**

| | |
|---|---|
| *type* | Code type |
| *i* | Codeword index |

**Returns**

*i-th* codeword of the code *type*

### 7.2.5 Friends And Related Function Documentation

#### 7.2.5.1 internal::Singleton< const Codes >

```
friend class internal::Singleton< const Codes > [friend]
```

The documentation for this class was generated from the following file:

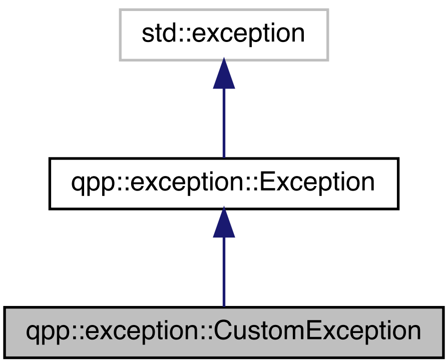
- classes/codes.h

## 7.3 qpp::exception::CustomException Class Reference

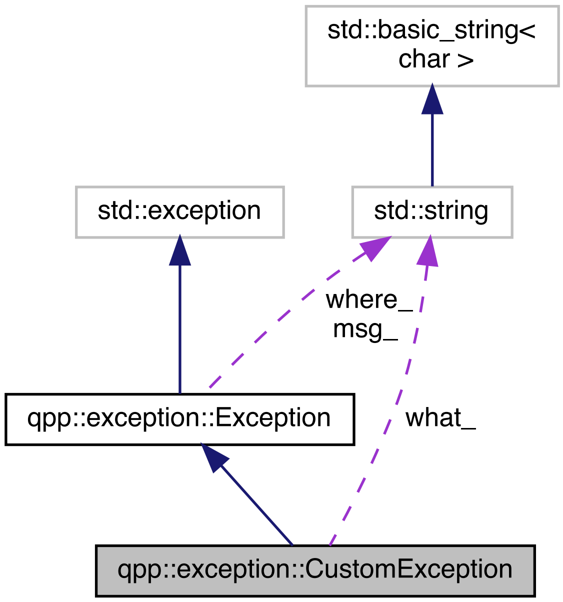
Custom exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::CustomException:



Collaboration diagram for qpp::exception::CustomException:




## Public Member Functions

- CustomException (const std::string &where, const std::string &what)

## Private Member Functions

- std::string type_description () const override

  *Exception type description.*

## Private Attributes

- std::string what_ {}

### 7.3.1 Detailed Description

Custom exception.

Custom exception, the user must provide a custom message

### 7.3.2 Constructor & Destructor Documentation

#### 7.3.2.1 CustomException()

```
qpp::exception::CustomException::CustomException (
            const std::string & where,
            const std::string & what )  [inline]
```

### 7.3.3 Member Function Documentation

#### 7.3.3.1 type_description()

```
std::string qpp::exception::CustomException::type_description ( ) const  [inline], [override],
[private], [virtual]
```

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

### 7.3.4 Member Data Documentation

#### 7.3.4.1 what_

```
std::string qpp::exception::CustomException::what_ {}  [private]
```

The documentation for this class was generated from the following file:

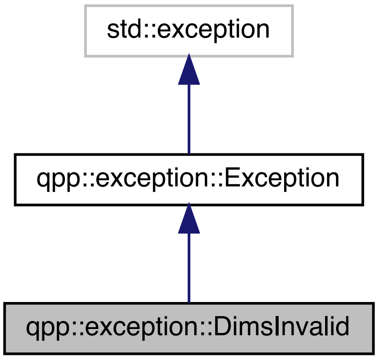
- classes/exception.h

## 7.4 qpp::exception::DimsInvalid Class Reference

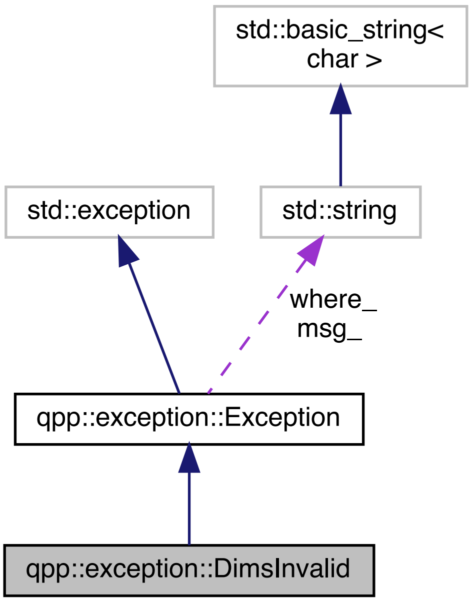
Invalid dimension(s) exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::DimsInvalid:




Collaboration diagram for qpp::exception::DimsInvalid:



## Public Member Functions

- std::string type_description () const override

    *Exception type description.*
- Exception (const std::string &where)

    *Constructs an exception.*

### 7.4.1 Detailed Description

Invalid dimension(s) exception.

std::vector<idx> of dimensions has zero size or contains zeros

### 7.4.2 Member Function Documentation

#### 7.4.2.1 Exception()

`qpp::exception::Exception::Exception  [inline], [explicit]`

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.4.2.2 type_description()

`std::string qpp::exception::DimsInvalid::type_description ( ) const  [inline], [override], [virtual]`

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

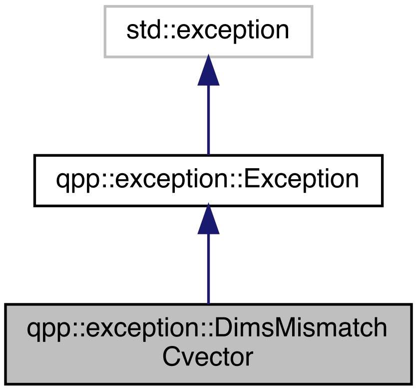
- classes/exception.h

## 7.5 qpp::exception::DimsMismatchCvector Class Reference

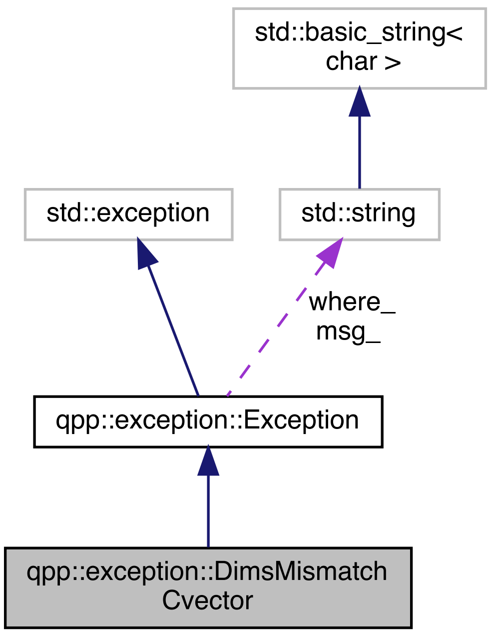
Dimension(s) mismatch column vector size exception.

`#include <classes/exception.h>`

Inheritance diagram for qpp::exception::DimsMismatchCvector:




Collaboration diagram for qpp::exception::DimsMismatchCvector:



### Public Member Functions

- std::string type_description () const override
  *Exception type description.*
- Exception (const std::string &where)
  *Constructs an exception.*

## 7.5.1 Detailed Description

Dimension(s) mismatch column vector size exception.

Product of the elements of std::vector<idx> of dimensions is not equal to the number of elements of the Eigen::↩
Matrix (assumed to be a column vector)

## 7.5.2 Member Function Documentation

### 7.5.2.1 Exception()

`qpp::exception::Exception::Exception [inline], [explicit]`

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

### 7.5.2.2 type_description()

`std::string qpp::exception::DimsMismatchCvector::type_description ( ) const [inline], [override], [virtual]`

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

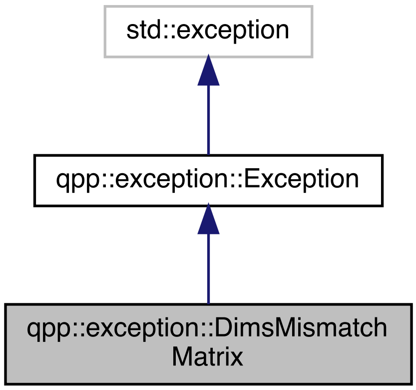
- classes/exception.h

## 7.6 qpp::exception::DimsMismatchMatrix Class Reference

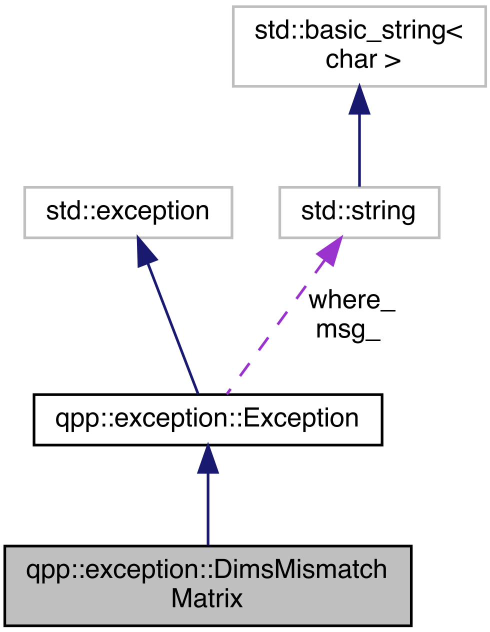
Dimension(s) mismatch matrix size exception.

`#include <classes/exception.h>`

Inheritance diagram for qpp::exception::DimsMismatchMatrix:




Collaboration diagram for qpp::exception::DimsMismatchMatrix:



### Public Member Functions

- std::string type_description () const override
  *Exception type description.*
- Exception (const std::string &where)
  *Constructs an exception.*

## 7.6.1 Detailed Description

Dimension(s) mismatch matrix size exception.

Product of the elements of std::vector<idx> of dimensions is not equal to the number of rows of the Eigen::Matrix (assumed to be a square matrix)

## 7.6.2 Member Function Documentation

### 7.6.2.1 Exception()

```
qpp::exception::Exception::Exception  [inline], [explicit]
```

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

### 7.6.2.2 type_description()

```
std::string qpp::exception::DimsMismatchMatrix::type_description ( ) const  [inline], [override],
[virtual]
```

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

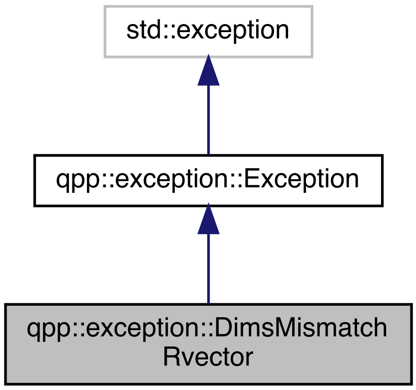
- classes/exception.h

## 7.7 qpp::exception::DimsMismatchRvector Class Reference

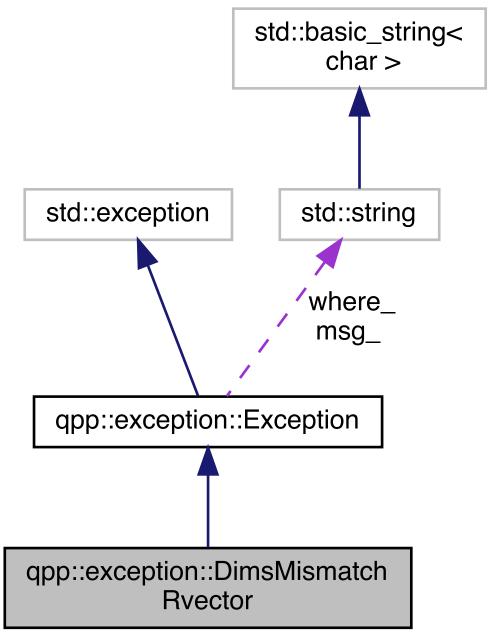
Dimension(s) mismatch row vector size exception.

`#include <classes/exception.h>`

Inheritance diagram for qpp::exception::DimsMismatchRvector:




Collaboration diagram for qpp::exception::DimsMismatchRvector:



## Public Member Functions

- std::string type_description () const override

  *Exception type description.*
- Exception (const std::string &where)

  *Constructs an exception.*

### 7.7.1 Detailed Description

Dimension(s) mismatch row vector size exception.

Product of the elements of std::vector<idx> of dimensions is not equal to the number of elements of the Eigen::↩
Matrix (assumed to be a row vector)

### 7.7.2 Member Function Documentation

#### 7.7.2.1 Exception()

`qpp::exception::Exception::Exception [inline], [explicit]`

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.7.2.2 type_description()

`std::string qpp::exception::DimsMismatchRvector::type_description ( ) const [inline], [override], [virtual]`

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

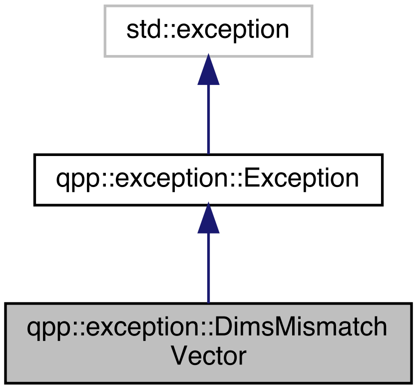
- classes/exception.h

## 7.8 qpp::exception::DimsMismatchVector Class Reference

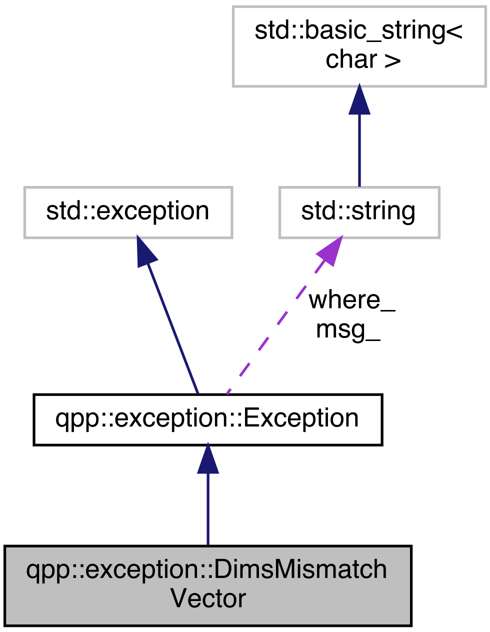
Dimension(s) mismatch vector size exception.

`#include <classes/exception.h>`

Inheritance diagram for qpp::exception::DimsMismatchVector:




Collaboration diagram for qpp::exception::DimsMismatchVector:



### Public Member Functions

- std::string type_description () const override

  *Exception type description.*
- Exception (const std::string &where)

  *Constructs an exception.*

## 7.8.1 Detailed Description

Dimension(s) mismatch vector size exception.

Product of the elements of std::vector<idx> of dimensions is not equal to the number of elements of the Eigen::↩ Matrix (assumed to be a row/column vector)

## 7.8.2 Member Function Documentation

### 7.8.2.1 Exception()

`qpp::exception::Exception::Exception [inline], [explicit]`

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

### 7.8.2.2 type_description()

`std::string qpp::exception::DimsMismatchVector::type_description ( ) const [inline], [override], [virtual]`

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

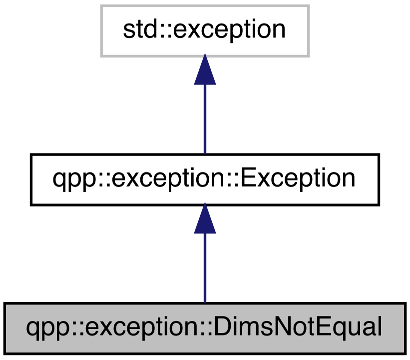
- classes/exception.h

## 7.9 qpp::exception::DimsNotEqual Class Reference

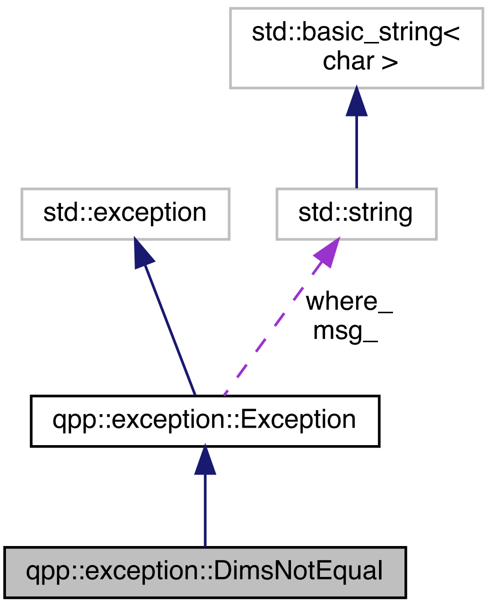
Dimensions not equal exception.

`#include <classes/exception.h>`

Inheritance diagram for qpp::exception::DimsNotEqual:




Collaboration diagram for qpp::exception::DimsNotEqual:



## Public Member Functions

- std::string type_description () const override
  *Exception type description.*
- Exception (const std::string &where)
  *Constructs an exception.*

### 7.9.1 Detailed Description

Dimensions not equal exception.

Local/global dimensions are not equal

### 7.9.2 Member Function Documentation

#### 7.9.2.1 Exception()

```
qpp::exception::Exception::Exception  [inline], [explicit]
```

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.9.2.2 type_description()

```
std::string qpp::exception::DimsNotEqual::type_description ( ) const  [inline], [override],
[virtual]
```

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

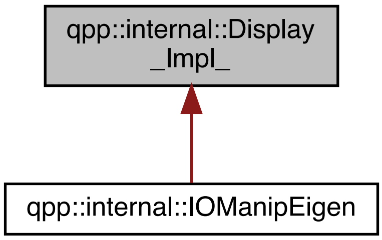
The documentation for this class was generated from the following file:

- classes/exception.h

## 7.10 qpp::internal::Display_Impl_ Struct Reference

`#include <internal/util.h>`

Inheritance diagram for qpp::internal::Display_Impl_:




### Public Member Functions

- template< typename T >
  std::ostream & display_impl_ (const T &A, std::ostream &os, double chop=qpp::chop) const

### 7.10.1 Member Function Documentation

#### 7.10.1.1 display_impl_()

```
template<typename T >
std::ostream& qpp::internal::Display_Impl_::display_impl_ (
            const T & A,
            std::ostream & os,
            double chop = qpp::chop ) const  [inline]
```

The documentation for this struct was generated from the following file:

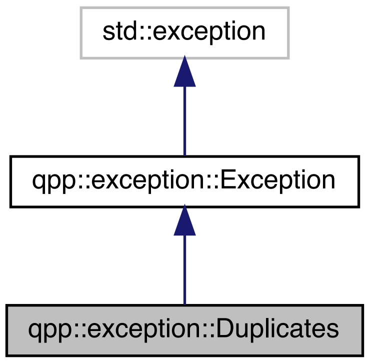
- internal/util.h

## 7.11 qpp::exception::Duplicates Class Reference

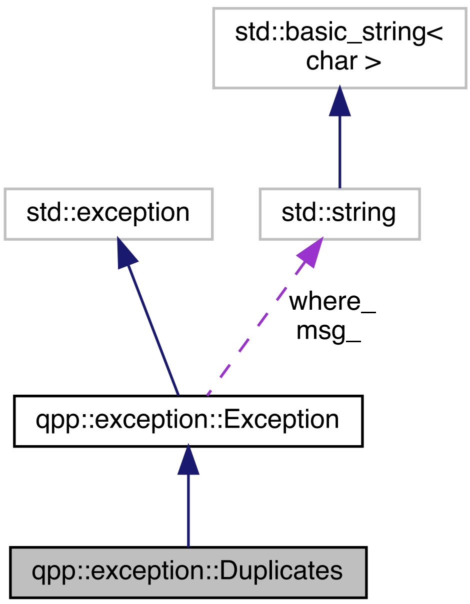
System (e.g. std::vector) has duplicates exception.

`#include <classes/exception.h>`

Inheritance diagram for qpp::exception::Duplicates:




Collaboration diagram for qpp::exception::Duplicates:



## Public Member Functions

- std::string type_description () const override

  *Exception type description.*
- Exception (const std::string &where)

  *Constructs an exception.*

### 7.11.1 Detailed Description

System (e.g. std::vector) has duplicates exception.

### 7.11.2 Member Function Documentation

#### 7.11.2.1 Exception()

```
qpp::exception::Exception::Exception  [inline], [explicit]
```

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.11.2.2 type_description()

```
std::string qpp::exception::Duplicates::type_description ( ) const  [inline], [override],
[virtual]
```

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

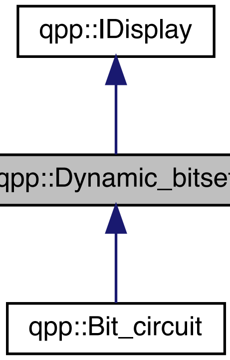
- classes/exception.h

## 7.12 qpp::Dynamic_bitset Class Reference

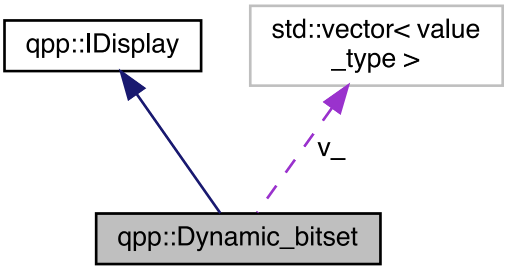
Dynamic bitset class, allows the specification of the number of bits at runtime.

```
#include <classes/reversible.h>
```

Inheritance diagram for qpp::Dynamic_bitset:




Collaboration diagram for qpp::Dynamic_bitset:




### Public Types

- using value_type = unsigned int

    *type of the storage elements*
- using storage_type = std::vector< value_type >

    *type of the storage*

## Public Member Functions

- Dynamic_bitset (idx n)

  *Constructor, initializes all bits to false (zero)*
- virtual ~Dynamic_bitset ()=default

  *Default virtual destructor.*
- const storage_type & data () const

  *Raw storage space of the bitset.*
- idx size () const noexcept

  *Number of bits stored in the bitset.*
- idx storage_size () const noexcept

  *Size of the underlying storage space (in units of value_type, unsigned int by default)*
- idx count () const noexcept

  *Number of bits set to one in the bitset (Hamming weight)*
- bool get (idx pos) const noexcept

  *The value of the bit at position pos.*
- bool none () const noexcept

  *Checks whether none of the bits are set.*
- bool all () const noexcept

  *Checks whether all bits are set.*
- bool any () const noexcept

  *Checks whether any bit is set.*
- Dynamic_bitset & set (idx pos, bool value=true)

  *Sets the bit at position pos.*
- Dynamic_bitset & set () noexcept

  *Set all bits to true.*
- Dynamic_bitset & rand (idx pos, double p=0.5)

  *Sets the bit at position pos according to a Bernoulli(p) distribution.*
- Dynamic_bitset & rand (double p=0.5)

  *Sets all bits according to a Bernoulli(p) distribution.*
- Dynamic_bitset & reset (idx pos)

  *Sets the bit at position pos to false.*
- Dynamic_bitset & reset () noexcept

  *Sets all bits to false.*
- Dynamic_bitset & flip (idx pos)

  *Flips the bit at position pos.*
- Dynamic_bitset & flip () noexcept

  *Flips all bits.*
- bool operator== (const Dynamic_bitset &rhs) const noexcept

  *Equality operator.*
- bool operator!= (const Dynamic_bitset &rhs) const noexcept

  *Inequality operator.*
- idx operator- (const Dynamic_bitset &rhs) const noexcept

  *Number of places the two bitsets differ (Hamming distance)*
- template<class CharT = char, class Traits = std::char_traits<CharT>, class Allocator = std::allocator<CharT>>
  std::basic_string< CharT, Traits, Allocator > to_string (CharT zero=CharT('0'), CharT one=CharT('1')) const

  *String representation.*

## Protected Member Functions

- idx index_ (idx pos) const
  *Index of the pos bit in the storage space.*
- idx offset_ (idx pos) const
  *Offset of the pos bit in the storage space relative to its index.*

## Protected Attributes

- idx storage_size_
  *storage size*
- idx n_
  *number of bits*
- std::vector< value_type > v_
  *storage space*

## Private Member Functions

- std::ostream & display (std::ostream &os) const override
  *qpp::IDisplay::display() override, displays the bitset bit by bit*

### 7.12.1 Detailed Description

Dynamic bitset class, allows the specification of the number of bits at runtime.

**Note**

The interface mimics std::bitset<>

### 7.12.2 Member Typedef Documentation

#### 7.12.2.1 storage_type

```
using qpp::Dynamic_bitset::storage_type = std::vector<value_type>
```

type of the storage

#### 7.12.2.2 value_type

```
using qpp::Dynamic_bitset::value_type = unsigned int
```

type of the storage elements

### 7.12.3 Constructor & Destructor Documentation

#### 7.12.3.1 Dynamic_bitset()

```
qpp::Dynamic_bitset::Dynamic_bitset (
            idx n )  [inline], [explicit]
```

Constructor, initializes all bits to false (zero)

**Parameters**

| | |
|---|---|
| *n* | Number of bits in the bitset |

#### 7.12.3.2 ∼Dynamic_bitset()

```
virtual qpp::Dynamic_bitset::∼Dynamic_bitset ( ) [virtual], [default]
```

Default virtual destructor.

### 7.12.4 Member Function Documentation

#### 7.12.4.1 all()

```
bool qpp::Dynamic_bitset::all ( ) const [inline], [noexcept]
```

Checks whether all bits are set.

**Returns**

True if all of the bits are set

#### 7.12.4.2 any()

```
bool qpp::Dynamic_bitset::any ( ) const [inline], [noexcept]
```

Checks whether any bit is set.

**Returns**

True if any of the bits is set

#### 7.12.4.3 count()

```
idx qpp::Dynamic_bitset::count ( ) const [inline], [noexcept]
```

Number of bits set to one in the bitset (Hamming weight)

**Returns**

Hamming weight

#### 7.12.4.4 data()

```
const storage_type& qpp::Dynamic_bitset::data ( ) const [inline]
```

Raw storage space of the bitset.

**Returns**

Const reference to the underlying storage space

#### 7.12.4.5 display()

```
std::ostream& qpp::Dynamic_bitset::display (
            std::ostream & os ) const [inline], [override], [private], [virtual]
```

qpp::IDisplay::display() override, displays the bitset bit by bit

**Parameters**

| | |
|---|---|
| *os* | Output stream passed by reference |

**Returns**

Reference to the output stream

Implements qpp::IDisplay.

#### 7.12.4.6 flip() [1/2]

```
Dynamic_bitset& qpp::Dynamic_bitset::flip (
            idx pos ) [inline]
```

Flips the bit at position *pos*.

**Parameters**

| | |
|---|---|
| *pos* | Position in the bitset |

**Returns**

Reference to the current instance

#### 7.12.4.7 flip() [2/2]

```
Dynamic_bitset& qpp::Dynamic_bitset::flip ( ) [inline], [noexcept]
```

Flips all bits.

**Returns**

Reference to the current instance

#### 7.12.4.8 get()

```
bool qpp::Dynamic_bitset::get (
            idx pos ) const [inline], [noexcept]
```

The value of the bit at position *pos*.

**Parameters**

| | |
|---|---|
| *pos* | Position in the bitset |

**Returns**

The value of the bit at position *pos*

#### 7.12.4.9 index_()

```
idx qpp::Dynamic_bitset::index_ (
            idx pos ) const [inline], [protected]
```

Index of the *pos* bit in the storage space.

**Parameters**

| | |
|---|---|
| *pos* | Bit location |

**Returns**

Index of the *pos* bit in the storage space

#### 7.12.4.10 none()

```
bool qpp::Dynamic_bitset::none ( ) const [inline], [noexcept]
```

Checks whether none of the bits are set.

**Returns**

True if none of the bits are set

#### 7.12.4.11 offset_()

```
idx qpp::Dynamic_bitset::offset_ (
            idx pos ) const [inline], [protected]
```

Offset of the *pos* bit in the storage space relative to its index.

**Parameters**

| | |
|---|---|
| *pos* | Bit location |

**Returns**

Offset of the *pos* bit in the storage space relative to its index

#### 7.12.4.12 operator"!=()

```
bool qpp::Dynamic_bitset::operator!= (
            const Dynamic_bitset & rhs ) const [inline], [noexcept]
```

Inequality operator.

**Parameters**

| | |
|---|---|
| *rhs* | Dynamic_bitset against which the inequality is being tested |

**Returns**

True if the bitsets are not equal (bit by bit), false otherwise

#### 7.12.4.13 operator-()

```
idx qpp::Dynamic_bitset::operator- (
            const Dynamic_bitset & rhs ) const  [inline], [noexcept]
```

Number of places the two bitsets differ (Hamming distance)

**Parameters**

| | |
|---|---|
| *rhs* | Dynamic_bitset against which the the Hamming distance is computed |

**Returns**

Hamming distance

#### 7.12.4.14 operator==()

```
bool qpp::Dynamic_bitset::operator== (
            const Dynamic_bitset & rhs ) const  [inline], [noexcept]
```

Equality operator.

**Parameters**

| | |
|---|---|
| *rhs* | Dynamic_bitset against which the equality is being tested |

**Returns**

True if the bitsets are equal (bit by bit), false otherwise

#### 7.12.4.15 rand() [1/2]

```
Dynamic_bitset& qpp::Dynamic_bitset::rand (
            idx pos,
            double p = 0.5 )  [inline]
```

Sets the bit at position *pos* according to a Bernoulli(p) distribution.

**Parameters**

| | |
|---|---|
| *pos* | Position in the bitset |
| *p* | Probability |

**Returns**

Reference to the current instance

#### 7.12.4.16 rand() [2/2]

```
Dynamic_bitset& qpp::Dynamic_bitset::rand (
            double p = 0.5 ) [inline]
```

Sets all bits according to a Bernoulli(p) distribution.

**Parameters**

| | |
|---|---|
| *p* | Probability |

**Returns**

Reference to the current instance

#### 7.12.4.17 reset() [1/2]

```
Dynamic_bitset& qpp::Dynamic_bitset::reset (
            idx pos ) [inline]
```

Sets the bit at position *pos* to false.

**Parameters**

| | |
|---|---|
| *pos* | Position in the bitset |

**Returns**

Reference to the current instance

#### 7.12.4.18 reset() [2/2]

```
Dynamic_bitset& qpp::Dynamic_bitset::reset ( ) [inline], [noexcept]
```

Sets all bits to false.

**Returns**

Reference to the current instance

#### 7.12.4.19 set() [1/2]

```
Dynamic_bitset& qpp::Dynamic_bitset::set (
            idx pos,
            bool value = true )  [inline]
```

Sets the bit at position *pos*.

**Parameters**

| | |
|---|---|
| *pos* | Position in the bitset |
| *value* | Bit value |

**Returns**

Reference to the current instance

#### 7.12.4.20 set() [2/2]

```
Dynamic_bitset& qpp::Dynamic_bitset::set ( )  [inline], [noexcept]
```

Set all bits to true.

**Returns**

Reference to the current instance

#### 7.12.4.21 size()

```
idx qpp::Dynamic_bitset::size ( ) const  [inline], [noexcept]
```

Number of bits stored in the bitset.

**Returns**

Number of bits stored in the bitset

#### 7.12.4.22 storage_size()

```
idx qpp::Dynamic_bitset::storage_size ( ) const  [inline], [noexcept]
```

Size of the underlying storage space (in units of value_type, unsigned int by default)

**Returns**

Size of the underlying storage space

#### 7.12.4.23 to_string()

```
template<class CharT = char, class Traits = std::char_traits<CharT>, class Allocator = std↵
::allocator<CharT>>
std::basic_string<CharT, Traits, Allocator> qpp::Dynamic_bitset::to_string (
            CharT zero = CharT('0'),
            CharT one = CharT('1') ) const  [inline]
```

String representation.

**Template Parameters**

| | |
|---|---|
| *CharT* | String character type |
| *Traits* | String traits |
| *Allocator* | String Allocator |

**Parameters**

| | |
|---|---|
| *zero* | Character representing the zero |
| *one* | Character representing the one |

**Returns**

The bitset as a string

### 7.12.5 Member Data Documentation

#### 7.12.5.1 n_

```
idx qpp::Dynamic_bitset::n_  [protected]
```

number of bits

#### 7.12.5.2 storage_size_

```
idx qpp::Dynamic_bitset::storage_size_  [protected]
```

storage size

#### 7.12.5.3 v_

```
std::vector<value_type> qpp::Dynamic_bitset::v_  [protected]
```

storage space

The documentation for this class was generated from the following file:

- classes/reversible.h

## 7.13 qpp::internal::EqualEigen Class Reference

Functor for comparing Eigen expressions for equality.

```
#include <functions.h>
```

### Public Member Functions

- template< typename Derived >
  bool operator() (const Eigen::MatrixBase< Derived > &A, const Eigen::MatrixBase< Derived > &B) const

### 7.13.1 Detailed Description

Functor for comparing Eigen expressions for equality.

**Note**

Works without assertion fails even if the dimensions of the arguments are different (in which case simply returns false

### 7.13.2 Member Function Documentation

#### 7.13.2.1 operator()()

```
template<typename Derived >
bool qpp::internal::EqualEigen::operator() (
            const Eigen::MatrixBase< Derived > & A,
            const Eigen::MatrixBase< Derived > & B ) const  [inline]
```

The documentation for this class was generated from the following file:

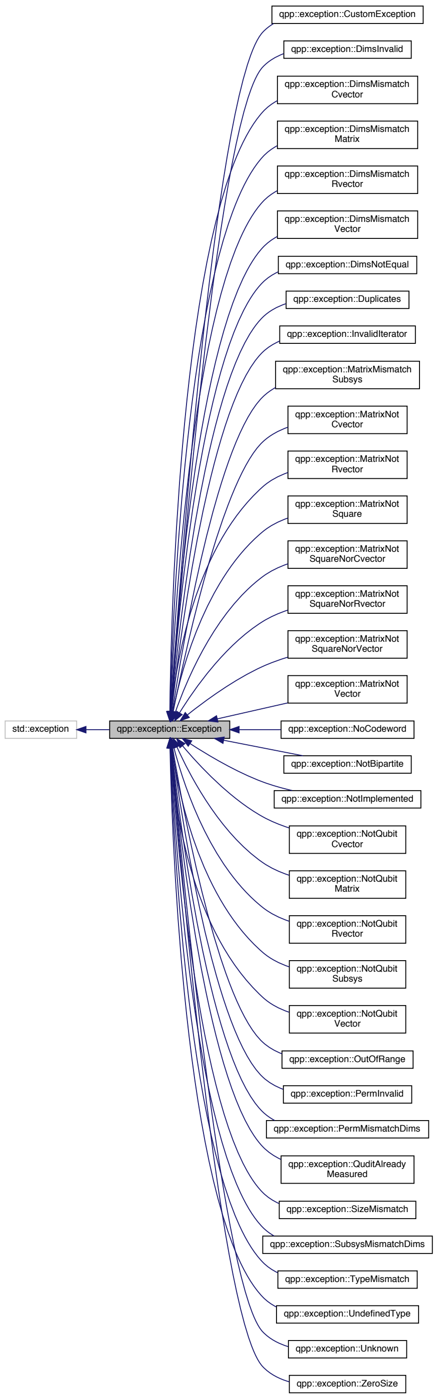
- functions.h

## 7.14 qpp::exception::Exception Class Reference

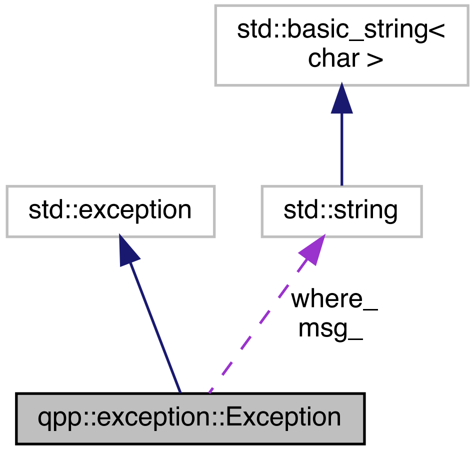
Base class for generating Quantum++ custom exceptions.

`#include <classes/exception.h>`

Inheritance diagram for qpp::exception::Exception:



Collaboration diagram for qpp::exception::Exception:




## Public Member Functions

- Exception (const std::string &where)

  *Constructs an exception.*
- virtual const char ∗ what () const noexcept override

  *Overrides std::exception::what()*
- virtual std::string type_description () const =0

  *Exception type description.*

## Private Attributes

- std::string where_
- std::string msg_

### 7.14.1 Detailed Description

Base class for generating Quantum++ custom exceptions.

Derive from this class if more exceptions are needed, making sure to override qpp::exception::Exception::type_description() in the derived class and to inherit the constructor qpp::exception::Exception::Exception(). Preferably keep your newly defined exception classes in the namespace qpp::exception.

Example:

```
namespace qpp
{
namespace exception
{
    class ZeroSize : public Exception
    {
    public:
        std::string type_description() const override
        {
            return "Object has zero size";
        }

        // inherit the base class' qpp::exception::Exception constructor
        using Exception::Exception;
    };
} // namespace exception
} // namespace qpp
```

## 7.14.2 Constructor & Destructor Documentation

### 7.14.2.1 Exception()

```
qpp::exception::Exception::Exception (
            const std::string & where )  [inline], [explicit]
```

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

## 7.14.3 Member Function Documentation

### 7.14.3.1 type_description()

```
std::string qpp::exception::Exception::type_description ( ) const  [inline], [pure virtual]
```

Exception type description.

**Returns**

Exception type description

Implemented in qpp::exception::InvalidIterator, qpp::exception::NotImplemented, qpp::exception::CustomException, qpp::exception::Duplicates, qpp::exception::QuditAlreadyMeasured, qpp::exception::UndefinedType, qpp::exception::SizeMismatch, qpp::exception::TypeMismatch, qpp::exception::OutOfRange, qpp::exception::NoCodeword, qpp::exception::NotBipartite, qpp::exception::NotQubitSubsys, qpp::exception::NotQubitVector, qpp::exception::NotQubitRvector, qpp::exception::NotQubitCvector, qpp::exception::NotQubitMatrix, qpp::exception::PermMismatchDims, qpp::exception::PermInvalid, qpp::exception::SubsysMismatchD qpp::exception::DimsMismatchVector, qpp::exception::DimsMismatchRvector, qpp::exception::DimsMismatchCvector, qpp::exception::DimsMismatchMatrix, qpp::exception::DimsNotEqual, qpp::exception::DimsInvalid, qpp::exception::MatrixMismatchSu qpp::exception::MatrixNotSquareNorVector, qpp::exception::MatrixNotSquareNorRvector, qpp::exception::MatrixNotSquareNorCvecto qpp::exception::MatrixNotVector, qpp::exception::MatrixNotRvector, qpp::exception::MatrixNotCvector, qpp::exception::MatrixNotSqua qpp::exception::ZeroSize, and qpp::exception::Unknown.

#### 7.14.3.2 what()

`virtual const char* qpp::exception::Exception::what ( ) const [inline], [override], [virtual], [noexcept]`

Overrides std::exception::what()

**Returns**

Exception description

### 7.14.4 Member Data Documentation

#### 7.14.4.1 msg_

`std::string qpp::exception::Exception::msg_ [mutable], [private]`

#### 7.14.4.2 where_

`std::string qpp::exception::Exception::where_ [private]`

The documentation for this class was generated from the following file:

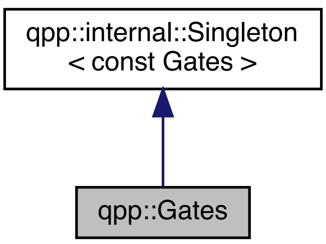
- classes/exception.h

## 7.15 qpp::Gates Class Reference

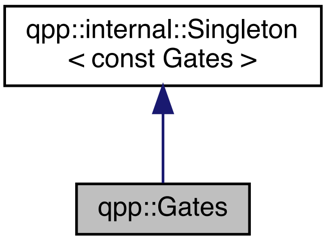
const Singleton class that implements most commonly used gates

`#include <classes/gates.h>`

Inheritance diagram for qpp::Gates:



Collaboration diagram for qpp::Gates:




### Public Member Functions

- cmat Rn (double theta, const std::vector< double > &n) const

  *Qubit rotation of theta about the 3-dimensional real (unit) vector n.*
- cmat RX (double theta) const

  *Qubit rotation of theta about the X axis.*
- cmat RY (double theta) const

  *Qubit rotation of theta about the Y axis.*
- cmat RZ (double theta) const

  *Qubit rotation of theta about the Z axis.*
- cmat Zd (idx D=2) const

  *Generalized Z gate for qudits.*
- cmat SWAPd (idx D=2) const

  *SWAP gate for qudits.*
- cmat Fd (idx D=2) const

  *Quantum Fourier transform gate for qudits.*
- cmat MODMUL (idx a, idx N, idx n) const

  *Modular multiplication gate for qubits Implements* $|x\rangle \longrightarrow |ax \bmod N\rangle$.
- cmat Xd (idx D=2) const

  *Generalized X gate for qudits.*
- template< typename Derived = Eigen::MatrixXcd>
  Derived Id (idx D=2) const

  *Identity gate.*
- template< typename Derived >
  dyn_mat< typename Derived::Scalar > CTRL (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &ctrl, const std::vector< idx > &target, idx n, idx d=2) const

  *Generates the multi-partite multiple-controlled-A gate in matrix form.*
- template< typename Derived >
  dyn_mat< typename Derived::Scalar > expandout (const Eigen::MatrixBase< Derived > &A, idx pos, const std::vector< idx > &dims) const

  *Expands out.*
- template< typename Derived >
  dyn_mat< typename Derived::Scalar > expandout (const Eigen::MatrixBase< Derived > &A, idx pos, const std::initializer_list< idx > &dims) const

  *Expands out.*

- template<typename Derived >
  dyn_mat< typename Derived::Scalar > expandout (const Eigen::MatrixBase< Derived > &A, idx pos, idx n, idx d=2) const

  *Expands out.*
- std::string get_name (const cmat &U) const

  *Get the name of the most common qubit gates.*

## Public Attributes

- cmat Id2 {cmat::Identity(2, 2)}

  *Identity gate.*
- cmat H {cmat::Zero(2, 2)}

  *Hadamard gate.*
- cmat X {cmat::Zero(2, 2)}

  *Pauli Sigma-X gate.*
- cmat Y {cmat::Zero(2, 2)}

  *Pauli Sigma-Y gate.*
- cmat Z {cmat::Zero(2, 2)}

  *Pauli Sigma-Z gate.*
- cmat S {cmat::Zero(2, 2)}

  *S gate.*
- cmat T {cmat::Zero(2, 2)}

  *T gate.*
- cmat CNOT {cmat::Identity(4, 4)}

  *Controlled-NOT control target gate.*
- cmat CZ {cmat::Identity(4, 4)}

  *Controlled-Phase gate.*
- cmat CNOTba {cmat::Zero(4, 4)}

  *Controlled-NOT target->control gate.*
- cmat SWAP {cmat::Identity(4, 4)}

  *SWAP gate.*
- cmat TOF {cmat::Identity(8, 8)}

  *Toffoli gate.*
- cmat FRED {cmat::Identity(8, 8)}

  *Fredkin gate.*

## Private Member Functions

- Gates ()

  *Initializes the gates.*
- ∼Gates ()=default

  *Default destructor.*

## Friends

- class internal::Singleton< const Gates >

## Additional Inherited Members

### 7.15.1 Detailed Description

const Singleton class that implements most commonly used gates

### 7.15.2 Constructor & Destructor Documentation

#### 7.15.2.1 Gates()

```
qpp::Gates::Gates ( ) [inline], [private]
```

Initializes the gates.

#### 7.15.2.2 ∼Gates()

```
qpp::Gates::~Gates ( ) [private], [default]
```

Default destructor.

### 7.15.3 Member Function Documentation

#### 7.15.3.1 CTRL()

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::Gates::CTRL (
            const Eigen::MatrixBase< Derived > & A,
            const std::vector< idx > & ctrl,
            const std::vector< idx > & target,
            idx n,
            idx d = 2 ) const [inline]
```

Generates the multi-partite multiple-controlled-*A* gate in matrix form.

**See also**

qpp::applyCTRL()

**Note**

The dimension of the gate *A* must match the dimension of *target*

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *ctrl* | Control subsystem indexes |
| *target* | Subsystem indexes where the gate *A* is applied |
| *n* | Total number of subsystems |
| *d* | Subsystem dimensions |

**Returns**

CTRL-A gate, as a matrix over the same scalar field as *A*

#### 7.15.3.2 expandout() [1/3]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::Gates::expandout (
            const Eigen::MatrixBase< Derived > & A,
            idx pos,
            const std::vector< idx > & dims ) const  [inline]
```

Expands out.

**See also**

qpp::kron()

Expands out *A* as a matrix in a multi-partite system. Faster than using qpp::kron(I, I, ..., I, A, I, ..., I).

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *pos* | Position |
| *dims* | Dimensions of the multi-partite system |

**Returns**

Tensor product $I \otimes \cdots \otimes I \otimes A \otimes I \otimes \cdots \otimes I$, with *A* on position *pos*, as a dynamic matrix over the same scalar field as *A*

#### 7.15.3.3 expandout() [2/3]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::Gates::expandout (
            const Eigen::MatrixBase< Derived > & A,
            idx pos,
            const std::initializer_list< idx > & dims ) const  [inline]
```

Expands out.

**See also**

qpp::kron()

Expands out *A* as a matrix in a multi-partite system. Faster than using qpp::kron(I, I, ..., I, A, I, ..., I).

**Note**

The std::initializer_list overload exists because otherwise, in the degenerate case when *dims* has only one element, the one element list is implicitly converted to the element's underlying type, i.e. qpp::idx, which has the net effect of picking the wrong (non-vector) qpp::expandout() overload

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *pos* | Position |
| *dims* | Dimensions of the multi-partite system |

**Returns**

Tensor product $I \otimes \cdots \otimes I \otimes A \otimes I \otimes \cdots \otimes I$, with *A* on position *pos*, as a dynamic matrix over the same scalar field as *A*

#### 7.15.3.4 expandout() [3/3]

```
template<typename Derived >
dyn_mat<typename Derived::Scalar> qpp::Gates::expandout (
            const Eigen::MatrixBase< Derived > & A,
            idx pos,
            idx n,
            idx d = 2 ) const  [inline]
```

Expands out.

**See also**

qpp::kron()

Expands out *A* as a matrix in a multi-partite system. Faster than using qpp::kron(I, I, ..., I, A, I, ..., I).

**Parameters**

| | |
|---|---|
| *A* | Eigen expression |
| *pos* | Position |
| *n* | Number of subsystems |
| *d* | Subsystem dimensions |

**Returns**

Tensor product $I \otimes \cdots \otimes I \otimes A \otimes I \otimes \cdots \otimes I$, with *A* on position *pos*, as a dynamic matrix over the same scalar field as *A*

#### 7.15.3.5 Fd()

```
cmat qpp::Gates::Fd (
            idx D = 2 ) const  [inline]
```

Quantum Fourier transform gate for qudits.

**Note**

Defined as $F = \sum_{j,k=0}^{D-1} \exp(2\pi \mathrm{i} jk/D)|j\rangle\langle k|$

**Parameters**

| | |
|---|---|
| *D* | Dimension of the Hilbert space |

**Returns**

Fourier transform gate for qudits

#### 7.15.3.6 get_name()

```
std::string qpp::Gates::get_name (
            const cmat & U ) const  [inline]
```

Get the name of the most common qubit gates.

**Note**

Assumes that the gate *U* is represented by a square matrix. If not, returns the empty string

**Parameters**

| | |
|---|---|
| *U* | Complex matrix representing the quantum gate |

**Returns**

The name of the gate (if any), otherwise the empty string

#### 7.15.3.7 Id()

```
template<typename Derived = Eigen::MatrixXcd>
Derived qpp::Gates::Id (
            idx D = 2 ) const  [inline]
```

Identity gate.

**Note**

Can change the return type from complex matrix (default) by explicitly specifying the template parameter

**Parameters**

| | |
|---|---|
| *D* | Dimension of the Hilbert space |

**Returns**

Identity gate on a Hilbert space of dimension *D*

#### 7.15.3.8 MODMUL()

```
cmat qpp::Gates::MODMUL (
            idx a,
            idx N,
            idx n ) const  [inline]
```

Modular multiplication gate for qubits Implements $|x\rangle \longrightarrow |ax \mathrm{mod} N\rangle$.

**Note**

For the gate to be unitary, *a* and *N* should be co-prime. The function does not check co-primality in release versions!
The number of qubits required to implement the gate should satisfy $n \geq \lceil \log_2(N) \rceil$

**Parameters**

| | |
|---|---|
| *a* | Positive integer less than *N* |
| *N* | Positive integer |
| *n* | Number of qubits required for implementing the gate |

**Returns**

Modular multiplication gate

#### 7.15.3.9 Rn()

```
cmat qpp::Gates::Rn (
            double theta,
            const std::vector< double > & n ) const  [inline]
```

Qubit rotation of *theta* about the 3-dimensional real (unit) vector *n*.

**Parameters**

| | |
|---|---|
| *theta* | Rotation angle |
| *n* | 3-dimensional real (unit) vector |

**Returns**

Rotation gate

#### 7.15.3.10 RX()

```
cmat qpp::Gates::RX (
            double theta ) const  [inline]
```

Qubit rotation of *theta* about the X axis.

**Parameters**

| | |
|---|---|
| *theta* | Rotation angle |

**Returns**

Rotation gate

#### 7.15.3.11 RY()

```
cmat qpp::Gates::RY (
            double theta ) const  [inline]
```

Qubit rotation of *theta* about the Y axis.

**Parameters**

| | |
|---|---|
| *theta* | Rotation angle |

**Returns**

Rotation gate

#### 7.15.3.12 RZ()

```
cmat qpp::Gates::RZ (
            double theta ) const  [inline]
```

Qubit rotation of *theta* about the Z axis.

**Parameters**

| | |
|---|---|
| *theta* | Rotation angle |

**Returns**

Rotation gate

#### 7.15.3.13 SWAPd()

```
cmat qpp::Gates::SWAPd (
            idx D = 2 ) const  [inline]
```

SWAP gate for qudits.

**Parameters**

| | |
|---|---|
| *D* | Dimension of the Hilbert space |

**Returns**

SWAP gate for qudits

#### 7.15.3.14 Xd()

```
cmat qpp::Gates::Xd (
            idx D = 2 ) const  [inline]
```

Generalized X gate for qudits.

**Note**

Defined as $X = \sum_{j=0}^{D-1} |j \oplus 1\rangle\langle j|$, i.e. raising operator $X|j\rangle = |j \oplus 1\rangle$

**Parameters**

| | |
|---|---|
| *D* | Dimension of the Hilbert space |

**Returns**

Generalized X gate for qudits

#### 7.15.3.15 Zd()

```
cmat qpp::Gates::Zd (
            idx D = 2 ) const [inline]
```

Generalized Z gate for qudits.

**Note**

Defined as $Z = \sum_{j=0}^{D-1} \exp(2\pi \mathrm{i} j/D) |j\rangle\langle j|$

**Parameters**

| | |
|---|---|
| *D* | Dimension of the Hilbert space |

**Returns**

Generalized Z gate for qudits

### 7.15.4 Friends And Related Function Documentation

#### 7.15.4.1 internal::Singleton< const Gates >

```
friend class internal::Singleton< const Gates > [friend]
```

### 7.15.5 Member Data Documentation

#### 7.15.5.1 CNOT

```
cmat qpp::Gates::CNOT {cmat::Identity(4, 4)}
```

Controlled-NOT control target gate.

#### 7.15.5.2 CNOTba

```
cmat qpp::Gates::CNOTba {cmat::Zero(4, 4)}
```

Controlled-NOT target->control gate.

#### 7.15.5.3 CZ

```
cmat qpp::Gates::CZ {cmat::Identity(4, 4)}
```

Controlled-Phase gate.

#### 7.15.5.4 FRED

```
cmat qpp::Gates::FRED {cmat::Identity(8, 8)}
```

Fredkin gate.

#### 7.15.5.5 H

```
cmat qpp::Gates::H {cmat::Zero(2, 2)}
```

Hadamard gate.

#### 7.15.5.6 Id2

```
cmat qpp::Gates::Id2 {cmat::Identity(2, 2)}
```

Identity gate.

#### 7.15.5.7 S

```
cmat qpp::Gates::S {cmat::Zero(2, 2)}
```

S gate.

#### 7.15.5.8 SWAP

```
cmat qpp::Gates::SWAP {cmat::Identity(4, 4)}
```

SWAP gate.

#### 7.15.5.9 T

```
cmat qpp::Gates::T {cmat::Zero(2, 2)}
```

T gate.

#### 7.15.5.10 TOF

```
cmat qpp::Gates::TOF {cmat::Identity(8, 8)}
```

Toffoli gate.

#### 7.15.5.11 X

```
cmat qpp::Gates::X {cmat::Zero(2, 2)}
```

Pauli Sigma-X gate.

#### 7.15.5.12 Y

```
cmat qpp::Gates::Y {cmat::Zero(2, 2)}
```

Pauli Sigma-Y gate.

#### 7.15.5.13 Z

```
cmat qpp::Gates::Z {cmat::Zero(2, 2)}
```

Pauli Sigma-Z gate.

The documentation for this class was generated from the following file:

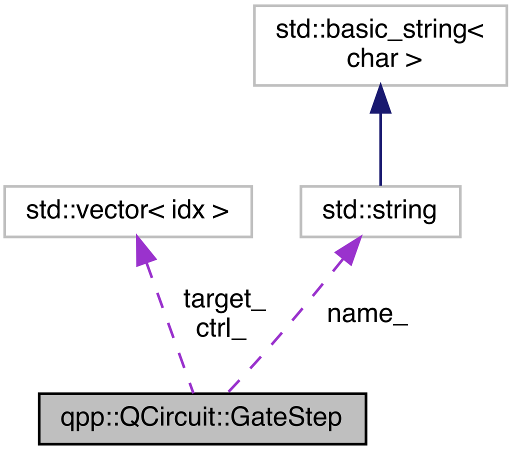
- classes/gates.h

## 7.16 qpp::QCircuit::GateStep Struct Reference

One step consisting only of gates/operators in the circuit.

`#include <classes/circuits/circuits.h>`

Collaboration diagram for qpp::QCircuit::GateStep:




### Public Member Functions

- GateStep ()=default
  *Default constructor.*
- GateStep (GateType gate_type, std::size_t gate_hash, const std::vector< idx > &ctrl, const std::vector< idx > &target, std::string name={})
  *Constructs a gate step instance.*

### Public Attributes

- GateType gate_type_ = GateType::NONE
  *gate type*
- std::size_t gate_hash_
  *gate hash*
- std::vector< idx > ctrl_
  *control*
- std::vector< idx > target_
  *target where the gate is applied*
- std::string name_
  *custom name of the step*

### 7.16.1 Detailed Description

One step consisting only of gates/operators in the circuit.

### 7.16.2 Constructor & Destructor Documentation

#### 7.16.2.1 GateStep() [1/2]

```
qpp::QCircuit::GateStep::GateStep ( ) [default]
```

Default constructor.

#### 7.16.2.2 GateStep() [2/2]

```
qpp::QCircuit::GateStep::GateStep (
            GateType gate_type,
            std::size_t gate_hash,
            const std::vector< idx > & ctrl,
            const std::vector< idx > & target,
            std::string name = {} ) [inline], [explicit]
```

Constructs a gate step instance.

**Parameters**

| | |
|---|---|
| *gate_type* | Gate type |
| *gate_hash* | Hash of the quantum gate |
| *ctrl* | Control qudit indexes |
| *target* | Target qudit indexes |
| *name* | Optional gate name |

### 7.16.3 Member Data Documentation

#### 7.16.3.1 ctrl_

```
std::vector<idx> qpp::QCircuit::GateStep::ctrl_
```

control

#### 7.16.3.2 gate_hash_

```
std::size_t qpp::QCircuit::GateStep::gate_hash_
```

gate hash

#### 7.16.3.3 gate_type_

```
GateType qpp::QCircuit::GateStep::gate_type_ = GateType::NONE
```

gate type

#### 7.16.3.4 name_

```
std::string qpp::QCircuit::GateStep::name_
```

custom name of the step

#### 7.16.3.5 target_

```
std::vector<idx> qpp::QCircuit::GateStep::target_
```

target where the gate is applied

The documentation for this struct was generated from the following file:

- classes/circuits/circuits.h

## 7.17 qpp::internal::HashEigen Class Reference

Functor for hashing Eigen expressions.

```
#include <functions.h>
```

### Public Member Functions

- template< typename Derived >
  std::size_t operator() (const Eigen::MatrixBase< Derived > &A) const

### 7.17.1 Detailed Description

Functor for hashing Eigen expressions.

### 7.17.2 Member Function Documentation

#### 7.17.2.1 operator()()

```
template<typename Derived >
std::size_t qpp::internal::HashEigen::operator() (
            const Eigen::MatrixBase< Derived > & A ) const  [inline]
```

The documentation for this class was generated from the following file:

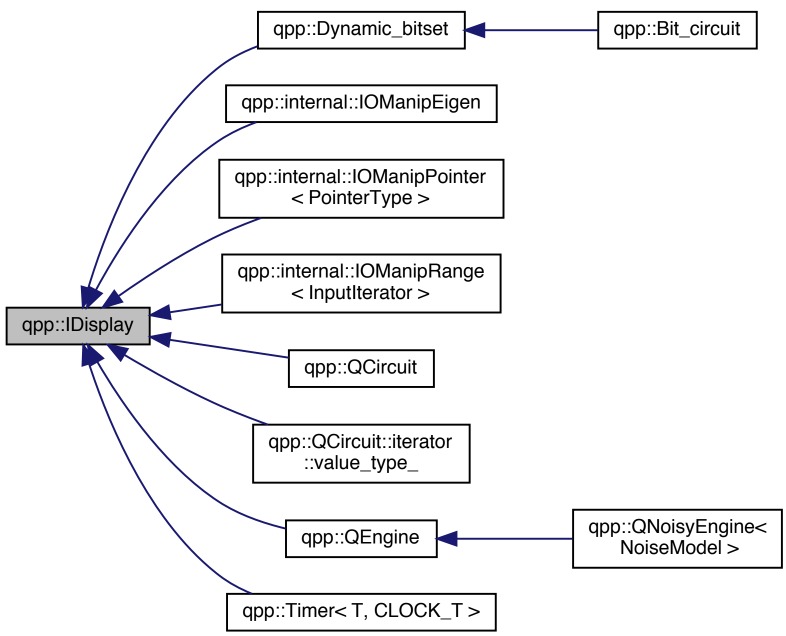
- functions.h

## 7.18 qpp::IDisplay Class Reference

Abstract class (interface) that mandates the definition of virtual std::ostream& display(std::ostream& os) const.

`#include <classes/idisplay.h>`

Inheritance diagram for qpp::IDisplay:



## Public Member Functions

- IDisplay ()=default

  *Default constructor.*
- IDisplay (const IDisplay &)=default

  *Default copy constructor.*
- IDisplay (IDisplay &&)=default

  *Default move constructor.*
- IDisplay & operator= (const IDisplay &)=default

  *Default copy assignment operator.*
- IDisplay & operator= (IDisplay &&)=default

  *Default move assignment operator.*
- virtual ∼IDisplay ()=default

  *Default virtual destructor.*

## Private Member Functions

- virtual std::ostream & display (std::ostream &os) const =0

  *Must be overridden by all derived classes.*

## Friends

- std::ostream & operator<< (std::ostream &os, const IDisplay &rhs)

  *Overloads the extraction operator.*

### 7.18.1 Detailed Description

Abstract class (interface) that mandates the definition of virtual std::ostream& display(std::ostream& os) const.

This class defines friend inline std::ostream& operator<< (std::ostream& os, const qpp::IDisplay& rhs). The latter delegates the work to the pure private virtual function qpp::IDisplay::display() which has to be overridden by all derived classes.

### 7.18.2 Constructor & Destructor Documentation

#### 7.18.2.1 IDisplay() [1/3]

```
qpp::IDisplay::IDisplay ( )  [default]
```

Default constructor.

#### 7.18.2.2 IDisplay() [2/3]

```
qpp::IDisplay::IDisplay (
            const IDisplay & ) [default]
```

Default copy constructor.

#### 7.18.2.3 IDisplay() [3/3]

```
qpp::IDisplay::IDisplay (
            IDisplay && ) [default]
```

Default move constructor.

#### 7.18.2.4 ∼IDisplay()

```
virtual qpp::IDisplay::∼IDisplay ( ) [virtual], [default]
```

Default virtual destructor.

### 7.18.3 Member Function Documentation

#### 7.18.3.1 display()

```
virtual std::ostream& qpp::IDisplay::display (
            std::ostream & os ) const [private], [pure virtual]
```

Must be overridden by all derived classes.

The actual stream extraction processing is performed by the overriden member function in the derived class. This function is automatically invoked by friend inline std::ostream& operator<<(std::ostream& os, const IDisplay& rhs).

Implemented in qpp::QCircuit, qpp::QEngine, qpp::QCircuit::iterator::value_type_, qpp::Dynamic_bitset, qpp::internal::IOManipEigen, qpp::Timer< T, CLOCK_T >, qpp::internal::IOManipPointer< PointerType >, and qpp::internal::IOManipRange< InputIterator >.

#### 7.18.3.2 operator=() [1/2]

```
IDisplay& qpp::IDisplay::operator= (
            const IDisplay & ) [default]
```

Default copy assignment operator.

#### 7.18.3.3 operator=() [2/2]

```
IDisplay& qpp::IDisplay::operator= (
            IDisplay && ) [default]
```

Default move assignment operator.

### 7.18.4 Friends And Related Function Documentation

#### 7.18.4.1 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const IDisplay & rhs ) [friend]
```

Overloads the extraction operator.

Delegates the work to the virtual function qpp::IDisplay::display()

The documentation for this class was generated from the following file:

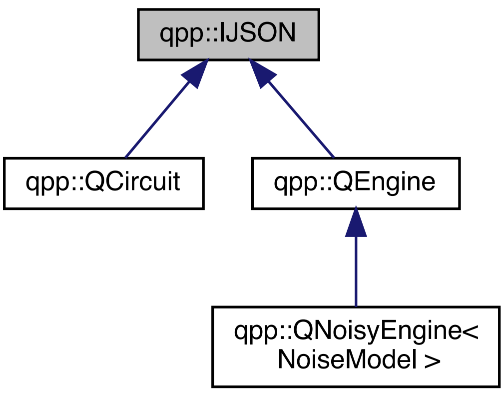
- classes/idisplay.h

## 7.19 qpp::IJSON Class Reference

Abstract class (interface) that mandates the definition of very basic JSON serialization support.

```
#include <classes/idisplay.h>
```

Inheritance diagram for qpp::IJSON:



## Public Member Functions

- IJSON ()=default
  *Default constructor.*
- IJSON (const IJSON &)=default
  *Default copy constructor.*
- IJSON (IJSON &&)=default
  *Default move constructor.*
- IJSON & operator= (const IJSON &)=default
  *Default copy assignment operator.*
- IJSON & operator= (IJSON &&)=default
  *Default move assignment operator.*
- virtual ∼IJSON ()=default
  *Default virtual destructor.*
- virtual std::string to_JSON (bool enclosed_in_curly_brackets=true) const =0
  *JSON representation of the derived instance, must be overridden by all derived classes.*

### 7.19.1 Detailed Description

Abstract class (interface) that mandates the definition of very basic JSON serialization support.

### 7.19.2 Constructor & Destructor Documentation

#### 7.19.2.1 IJSON() [1/3]

```
qpp::IJSON::IJSON ( )  [default]
```

Default constructor.

#### 7.19.2.2 IJSON() [2/3]

```
qpp::IJSON::IJSON (
            const IJSON & )  [default]
```

Default copy constructor.

#### 7.19.2.3 IJSON() [3/3]

```
qpp::IJSON::IJSON (
            IJSON && )  [default]
```

Default move constructor.

#### 7.19.2.4 ∼IJSON()

```
virtual qpp::IJSON::∼IJSON ( )  [virtual], [default]
```

Default virtual destructor.

### 7.19.3 Member Function Documentation

#### 7.19.3.1 operator=() [1/2]

```
IJSON& qpp::IJSON::operator= (
            const IJSON &  )  [default]
```

Default copy assignment operator.

#### 7.19.3.2 operator=() [2/2]

```
IJSON& qpp::IJSON::operator= (
            IJSON &&  )  [default]
```

Default move assignment operator.

#### 7.19.3.3 to_JSON()

```
virtual std::string qpp::IJSON::to_JSON (
            bool enclosed_in_curly_brackets = true ) const  [pure virtual]
```

JSON representation of the derived instance, must be overridden by all derived classes.

**Parameters**

| | |
|---|---|
| *enclosed_in_curly_brackets* | If true, encloses the result in curly brackets |

Implemented in qpp::QCircuit, and qpp::QEngine.

The documentation for this class was generated from the following file:

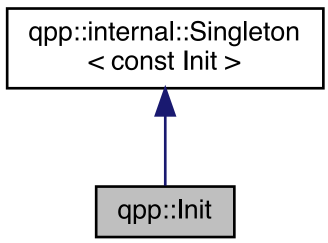
- classes/idisplay.h

## 7.20 qpp::Init Class Reference

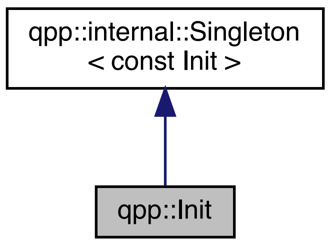
const Singleton class that performs additional initializations/cleanups

`#include <classes/init.h>`

Inheritance diagram for qpp::Init:




Collaboration diagram for qpp::Init:




### Private Member Functions

- Init ()
  *Additional initializations.*
- ∼Init ()
  *Cleanups.*

### Friends

- class internal::Singleton< const Init >

## Additional Inherited Members

### 7.20.1 Detailed Description

const Singleton class that performs additional initializations/cleanups

### 7.20.2 Constructor & Destructor Documentation

#### 7.20.2.1 Init()

`qpp::Init::Init ( ) [inline], [private]`

Additional initializations.

#### 7.20.2.2 ∼Init()

`qpp::Init::∼Init ( ) [inline], [private]`

Cleanups.

### 7.20.3 Friends And Related Function Documentation

#### 7.20.3.1 internal::Singleton< const Init >

`friend class internal::Singleton< const Init > [friend]`

The documentation for this class was generated from the following file:

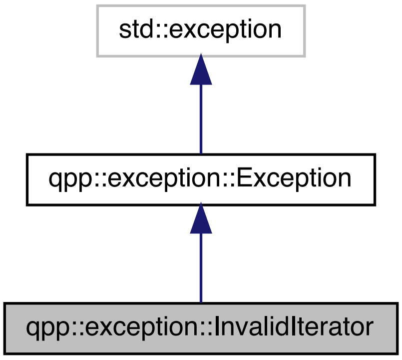
- classes/init.h

## 7.21 qpp::exception::InvalidIterator Class Reference

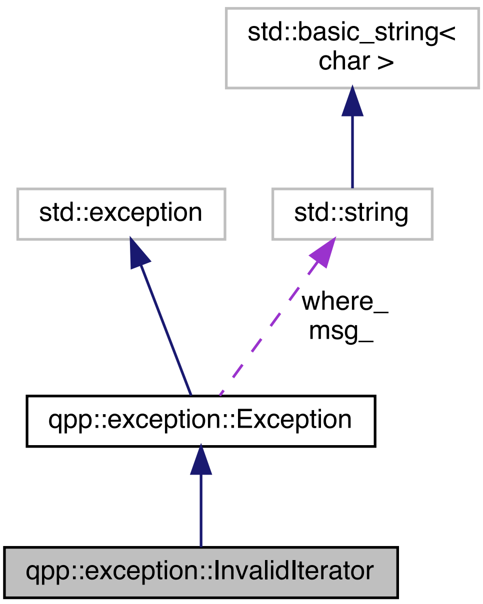
Invalid iterator.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::InvalidIterator:




Collaboration diagram for qpp::exception::InvalidIterator:



## Public Member Functions

- std::string type_description () const override

    *Exception type description.*
- Exception (const std::string &where)

    *Constructs an exception.*

### 7.21.1 Detailed Description

Invalid iterator.

### 7.21.2 Member Function Documentation

#### 7.21.2.1 Exception()

```
qpp::exception::Exception::Exception  [inline], [explicit]
```

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.21.2.2 type_description()

```
std::string qpp::exception::InvalidIterator::type_description ( ) const  [inline], [override],
[virtual]
```

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

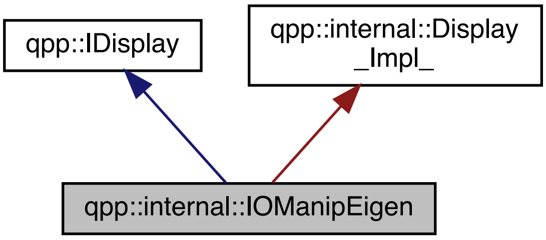
The documentation for this class was generated from the following file:

- classes/exception.h

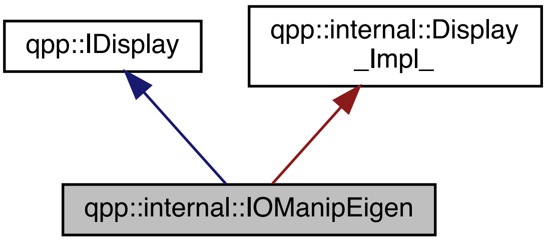
## 7.22 qpp::internal::IOManipEigen Class Reference

`#include <internal/classes/iomanip.h>`

Inheritance diagram for qpp::internal::IOManipEigen:




Collaboration diagram for qpp::internal::IOManipEigen:




### Public Member Functions

- template< typename Derived >
  IOManipEigen (const Eigen::MatrixBase< Derived > &A, double chop=qpp::chop)
- IOManipEigen (const cplx z, double chop=qpp::chop)

### Private Member Functions

- std::ostream & display (std::ostream &os) const override
  *Must be overridden by all derived classes.*

### Private Attributes

- cmat A_
- double chop_

## 7.22.1 Constructor & Destructor Documentation

### 7.22.1.1 IOManipEigen() [1/2]

```
template<typename Derived >
qpp::internal::IOManipEigen::IOManipEigen (
            const Eigen::MatrixBase< Derived > & A,
            double chop = qpp::chop )  [inline], [explicit]
```

### 7.22.1.2 IOManipEigen() [2/2]

```
qpp::internal::IOManipEigen::IOManipEigen (
            const cplx z,
            double chop = qpp::chop )  [inline], [explicit]
```

## 7.22.2 Member Function Documentation

### 7.22.2.1 display()

```
std::ostream& qpp::internal::IOManipEigen::display (
            std::ostream & os ) const  [inline], [override], [private], [virtual]
```

Must be overridden by all derived classes.

The actual stream extraction processing is performed by the overriden member function in the derived class. This function is automatically invoked by friend inline std::ostream& operator<<(std::ostream& os, const IDisplay& rhs).

Implements qpp::IDisplay.

## 7.22.3 Member Data Documentation

#### 7.22.3.1 A_

```
cmat qpp::internal::IOManipEigen::A_ [private]
```

#### 7.22.3.2 chop_

```
double qpp::internal::IOManipEigen::chop_ [private]
```

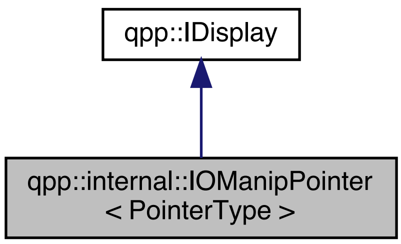
The documentation for this class was generated from the following file:

- internal/classes/iomanip.h

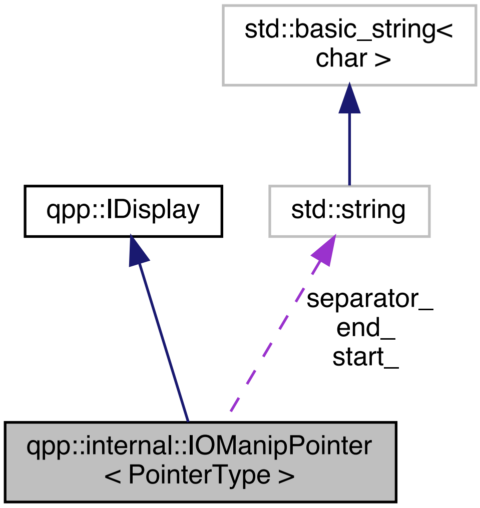
## 7.23 qpp::internal::IOManipPointer< PointerType > Class Template Reference

```
#include <internal/classes/iomanip.h>
```

Inheritance diagram for qpp::internal::IOManipPointer< PointerType >:



Collaboration diagram for qpp::internal::IOManipPointer< PointerType >:




## Public Member Functions

- IOManipPointer (const PointerType ∗p, idx N, const std::string &separator, const std::string &start="[", const std::string &end="]")
- IOManipPointer (const IOManipPointer &)=default
- IOManipPointer & operator= (const IOManipPointer &)=default

## Private Member Functions

- std::ostream & display (std::ostream &os) const override

    *Must be overridden by all derived classes.*

## Private Attributes

- const PointerType ∗ p_
- idx N_
- std::string separator_
- std::string start_
- std::string end_

### 7.23.1 Constructor & Destructor Documentation

#### 7.23.1.1 IOManipPointer() [1/2]

```
template<typename PointerType>
qpp::internal::IOManipPointer< PointerType >::IOManipPointer (
            const PointerType * p,
            idx N,
            const std::string & separator,
            const std::string & start = "[",
            const std::string & end = "]" )  [inline], [explicit]
```

#### 7.23.1.2 IOManipPointer() [2/2]

```
template<typename PointerType>
qpp::internal::IOManipPointer< PointerType >::IOManipPointer (
            const IOManipPointer< PointerType > & )  [default]
```

### 7.23.2 Member Function Documentation

#### 7.23.2.1 display()

```
template<typename PointerType>
std::ostream& qpp::internal::IOManipPointer< PointerType >::display (
            std::ostream & os ) const  [inline], [override], [private], [virtual]
```

Must be overridden by all derived classes.

The actual stream extraction processing is performed by the overriden member function in the derived class. This function is automatically invoked by friend inline std::ostream& operator<<(std::ostream& os, const IDisplay& rhs).

Implements qpp::IDisplay.

#### 7.23.2.2 operator=()

```
template<typename PointerType>
IOManipPointer& qpp::internal::IOManipPointer< PointerType >::operator= (
            const IOManipPointer< PointerType > & )  [default]
```

### 7.23.3 Member Data Documentation

#### 7.23.3.1 end_

```
template<typename PointerType>
std::string qpp::internal::IOManipPointer< PointerType >::end_  [private]
```

#### 7.23.3.2 N_

```
template<typename PointerType>
idx qpp::internal::IOManipPointer< PointerType >::N_  [private]
```

#### 7.23.3.3 p_

```
template<typename PointerType>
const PointerType* qpp::internal::IOManipPointer< PointerType >::p_  [private]
```

#### 7.23.3.4 separator_

```
template<typename PointerType>
std::string qpp::internal::IOManipPointer< PointerType >::separator_  [private]
```

#### 7.23.3.5 start_

```
template<typename PointerType>
std::string qpp::internal::IOManipPointer< PointerType >::start_  [private]
```

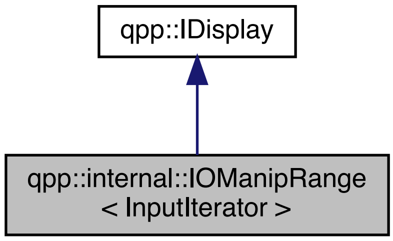
The documentation for this class was generated from the following file:

- internal/classes/iomanip.h

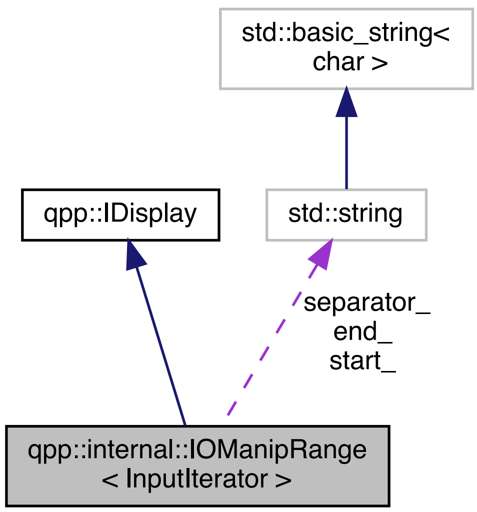
## 7.24 qpp::internal::IOManipRange< InputIterator > Class Template Reference

```
#include <internal/classes/iomanip.h>
```

Inheritance diagram for qpp::internal::IOManipRange< InputIterator >:




Collaboration diagram for qpp::internal::IOManipRange< InputIterator >:




### Public Member Functions

- IOManipRange (InputIterator first, InputIterator last, const std::string &separator, const std::string &start="[", const std::string &end="]")
- IOManipRange (const IOManipRange &)=default
- IOManipRange & operator= (const IOManipRange &)=default

### Private Member Functions

- std::ostream & display (std::ostream &os) const override
  *Must be overridden by all derived classes.*

### Private Attributes

- InputIterator first_
- InputIterator last_
- std::string separator_
- std::string start_
- std::string end_

## 7.24.1 Constructor & Destructor Documentation

### 7.24.1.1 IOManipRange() [1/2]

```
template<typename InputIterator>
qpp::internal::IOManipRange< InputIterator >::IOManipRange (
            InputIterator first,
            InputIterator last,
            const std::string & separator,
            const std::string & start = "[",
            const std::string & end = "]" )  [inline], [explicit]
```

### 7.24.1.2 IOManipRange() [2/2]

```
template<typename InputIterator>
qpp::internal::IOManipRange< InputIterator >::IOManipRange (
            const IOManipRange< InputIterator > &  )  [default]
```

## 7.24.2 Member Function Documentation

### 7.24.2.1 display()

```
template<typename InputIterator>
std::ostream& qpp::internal::IOManipRange< InputIterator >::display (
            std::ostream & os ) const  [inline], [override], [private], [virtual]
```

Must be overridden by all derived classes.

The actual stream extraction processing is performed by the overriden member function in the derived class. This function is automatically invoked by friend inline std::ostream& operator<<(std::ostream& os, const IDisplay& rhs).

Implements qpp::IDisplay.

#### 7.24.2.2 operator=()

```
template<typename InputIterator>
IOManipRange& qpp::internal::IOManipRange< InputIterator >::operator= (
            const IOManipRange< InputIterator > & ) [default]
```

### 7.24.3 Member Data Documentation

#### 7.24.3.1 end_

```
template<typename InputIterator>
std::string qpp::internal::IOManipRange< InputIterator >::end_ [private]
```

#### 7.24.3.2 first_

```
template<typename InputIterator>
InputIterator qpp::internal::IOManipRange< InputIterator >::first_ [private]
```

#### 7.24.3.3 last_

```
template<typename InputIterator>
InputIterator qpp::internal::IOManipRange< InputIterator >::last_ [private]
```

#### 7.24.3.4 separator_

```
template<typename InputIterator>
std::string qpp::internal::IOManipRange< InputIterator >::separator_ [private]
```

#### 7.24.3.5 start_

```
template<typename InputIterator>
std::string qpp::internal::IOManipRange< InputIterator >::start_ [private]
```

The documentation for this class was generated from the following file:

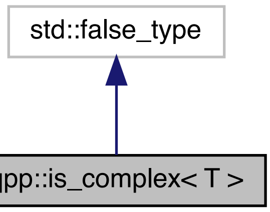
- internal/classes/iomanip.h

## 7.25 qpp::is_complex< T > Struct Template Reference

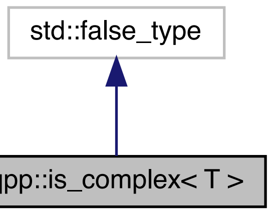
Checks whether the type is a complex type.

```
#include <traits.h>
```

Inheritance diagram for qpp::is_complex< T >:




Collaboration diagram for qpp::is_complex< T >:




### 7.25.1 Detailed Description

**template<typename T>**
**struct qpp::is_complex< T >**

Checks whether the type is a complex type.

Provides the constant member *value* which is equal to *true*, if the type is a complex type, i.e. *std::complex<T>*

The documentation for this struct was generated from the following file:

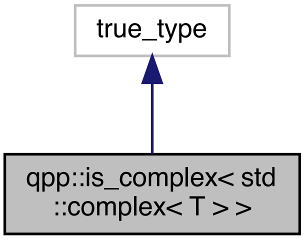
- traits.h

## 7.26 qpp::is_complex< std::complex< T > > Struct Template Reference

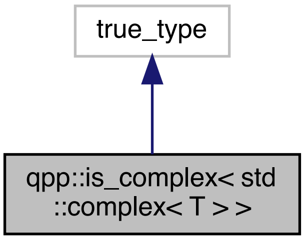
Checks whether the type is a complex number type, specialization for complex types.

```
#include <traits.h>
```

Inheritance diagram for qpp::is_complex< std::complex< T > >:




Collaboration diagram for qpp::is_complex< std::complex< T > >:




### 7.26.1 Detailed Description

**template<typename T>**
**struct qpp::is_complex< std::complex< T > >**

Checks whether the type is a complex number type, specialization for complex types.

The documentation for this struct was generated from the following file:

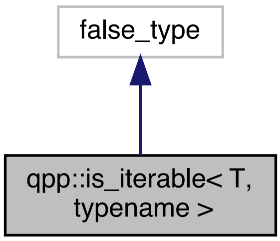
- traits.h

## 7.27 qpp::is_iterable< T, typename > Struct Template Reference

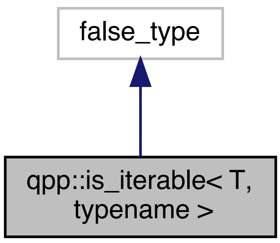
Checks whether *T* is compatible with an STL-like iterable container.

```
#include <traits.h>
```

Inheritance diagram for qpp::is_iterable< T, typename >:




Collaboration diagram for qpp::is_iterable< T, typename >:




### 7.27.1 Detailed Description

**template< typename T, typename = void >**
**struct qpp::is_iterable< T, typename >**

Checks whether *T* is compatible with an STL-like iterable container.

Provides the constant member *value* which is equal to *true*, if *T* is compatible with an iterable container, i.e. provides at least *begin()* and *end()* member functions and allows de-referencing. Otherwise, *value* is equal to *false*.

The documentation for this struct was generated from the following file:

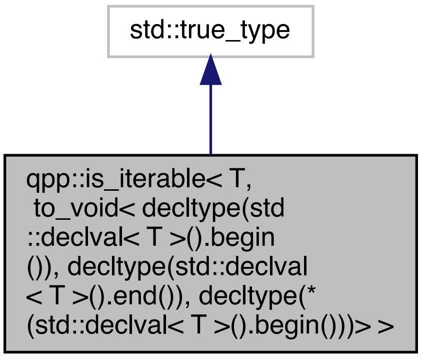
- traits.h

## 7.28 qpp::is_iterable< T, to_void< decltype(std::declval< T >().begin()), decltype(std::declval< T >().end()), decltype(∗(std::declval< T >().begin()))> > Struct Template Reference

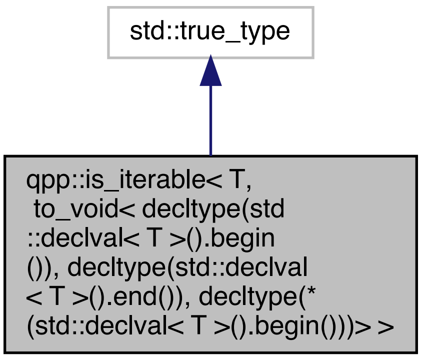
Checks whether *T* is compatible with an STL-like iterable container, specialization for STL-like iterable containers.

```
#include <traits.h>
```

Inheritance diagram for qpp::is_iterable< T, to_void< decltype(std::declval< T >().begin()), decltype(std::declval< T >().end()), decltype(∗(std::declval< T >().begin()))> >:




Collaboration diagram for qpp::is_iterable< T, to_void< decltype(std::declval< T >().begin()), decltype(std::declval< T >().end()), decltype(∗(std::declval< T >().begin()))> >:



### 7.28.1 Detailed Description

**template<typename T>**
**struct qpp::is_iterable< T, to_void< decltype(std::declval< T >().begin()), decltype(std::declval< T >().end()), decltype(∗(std::declval< T >().begin()))> >**

Checks whether *T* is compatible with an STL-like iterable container, specialization for STL-like iterable containers.

The documentation for this struct was generated from the following file:

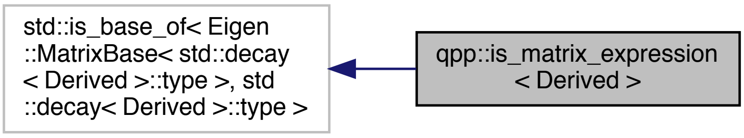
- traits.h

## 7.29 qpp::is_matrix_expression< Derived > Struct Template Reference

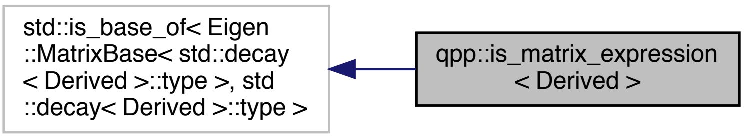
Checks whether the type is an Eigen matrix expression.

`#include <traits.h>`

Inheritance diagram for qpp::is_matrix_expression< Derived >:




Collaboration diagram for qpp::is_matrix_expression< Derived >:




### 7.29.1 Detailed Description

**template<typename Derived>**
**struct qpp::is_matrix_expression< Derived >**

Checks whether the type is an Eigen matrix expression.

Provides the constant member *value* which is equal to *true*, if the type is an Eigen matrix expression of type *Eigen::MatrixBase<Derived>*. Otherwise, *value* is equal to *false*.

The documentation for this struct was generated from the following file:

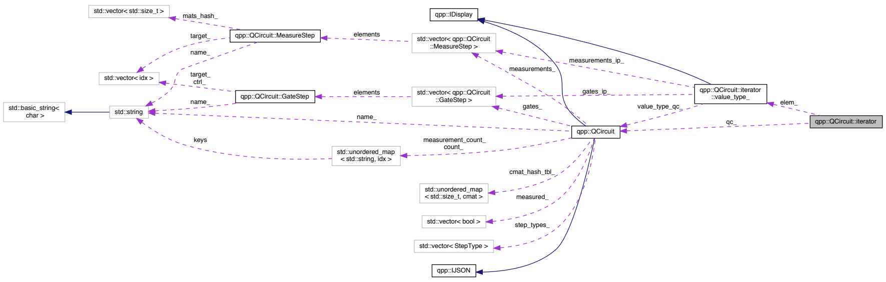
- traits.h

## 7.30 qpp::QCircuit::iterator Class Reference

Quantum circuit bound-checking (safe) iterator.

```
#include <classes/circuits/circuits.h>
```

Collaboration diagram for qpp::QCircuit::iterator:




### Classes

- class value_type_

    *Value type class for qpp::QCircuit::iterator.*

### Public Types

- using difference_type = long long

    *iterator trait*
- using value_type = value_type_

    *iterator trait*
- using pointer = const value_type ∗

    *iterator trait*
- using reference = const value_type &

    *iterator trait*
- using iterator_category = std::forward_iterator_tag

    *iterator trait*

### Public Member Functions

- iterator ()=default

    *Default constructor.*
- iterator (const iterator &)=default

    *Default copy constructor.*
- iterator & operator= (const iterator &)=default

    *Default copy assignment operator.*
- iterator & operator++ ()

    *Prefix increment operator.*
- iterator operator++ (int)

    *Postfix increment operator.*

- bool operator== (const iterator &rhs) const

    *Equality operator.*
- bool operator!= (iterator rhs) const

    *Inequality operator.*
- const value_type_ & operator∗ () const

    *Safe de-referencing operator.*
- void set_begin_ (const QCircuit ∗qc)

    *Sets the iterator to std::begin(this)*
- void set_end_ (const QCircuit ∗qc)

    *Sets the iterator to std::begin(this)*

## Private Attributes

- const QCircuit ∗ qc_ {nullptr}

    *< non-owning pointer to the parent const quantum circuit*
- value_type_ elem_ {nullptr}

### 7.30.1 Detailed Description

Quantum circuit bound-checking (safe) iterator.

**Note**

The iterator is a const_iterator by default

### 7.30.2 Member Typedef Documentation

#### 7.30.2.1 difference_type

```
using qpp::QCircuit::iterator::difference_type = long long
```

iterator trait

#### 7.30.2.2 iterator_category

```
using qpp::QCircuit::iterator::iterator_category = std::forward_iterator_tag
```

iterator trait

#### 7.30.2.3 pointer

```
using qpp::QCircuit::iterator::pointer = const value_type*
```

iterator trait

#### 7.30.2.4 reference

```
using qpp::QCircuit::iterator::reference = const value_type&
```

iterator trait

#### 7.30.2.5 value_type

```
using qpp::QCircuit::iterator::value_type = value_type_
```

iterator trait

### 7.30.3 Constructor & Destructor Documentation

#### 7.30.3.1 iterator() [1/2]

```
qpp::QCircuit::iterator::iterator ( )  [default]
```

Default constructor.

#### 7.30.3.2 iterator() [2/2]

```
qpp::QCircuit::iterator::iterator (
            const iterator & )  [default]
```

Default copy constructor.

### 7.30.4 Member Function Documentation

#### 7.30.4.1 operator"!=()

```
bool qpp::QCircuit::iterator::operator!= (
            iterator rhs ) const  [inline]
```

Inequality operator.

**Parameters**

| | |
|---|---|
| *rhs* | Iterator against which the inequality is being tested |

**Returns**

True if the iterators are not equal (bit by bit), false otherwise

#### 7.30.4.2 operator*()

```
const value_type_& qpp::QCircuit::iterator::operator* ( ) const [inline]
```

Safe de-referencing operator.

**Returns**

Constant reference to the iterator element

#### 7.30.4.3 operator++() [1/2]

```
iterator& qpp::QCircuit::iterator::operator++ ( ) [inline]
```

Prefix increment operator.

**Returns**

Reference to the current instance

#### 7.30.4.4 operator++() [2/2]

```
iterator qpp::QCircuit::iterator::operator++ (
            int ) [inline]
```

Postfix increment operator.

**Returns**

Copy of the current instance before the increment

#### 7.30.4.5 operator=()

```
iterator& qpp::QCircuit::iterator::operator= (
             const iterator & ) [default]
```

Default copy assignment operator.

**Returns**

Reference to the current instance

#### 7.30.4.6 operator==()

```
bool qpp::QCircuit::iterator::operator== (
             const iterator & rhs ) const [inline]
```

Equality operator.

**Parameters**

| | |
|---|---|
| *rhs* | Iterator against which the equality is being tested |

**Returns**

True if the iterators are equal, false otherwise

#### 7.30.4.7 set_begin_()

```
void qpp::QCircuit::iterator::set_begin_ (
             const QCircuit * qc ) [inline]
```

Sets the iterator to std::begin(this)

**Parameters**

| | |
|---|---|
| *qc* | Pointer to constant quantum circuit |

#### 7.30.4.8 set_end_()

```
void qpp::QCircuit::iterator::set_end_ (
             const QCircuit * qc ) [inline]
```

Sets the iterator to std::begin(this)

**Parameters**

| | |
|---|---|
| *qc* | Pointer to constant quantum circuit |

### 7.30.5 Member Data Documentation

#### 7.30.5.1 elem_

```
value_type_ qpp::QCircuit::iterator::elem_ {nullptr} [private]
```

#### 7.30.5.2 qc_

```
const QCircuit* qpp::QCircuit::iterator::qc_ {nullptr} [private]
```

< non-owning pointer to the parent const quantum circuit

The documentation for this class was generated from the following file:

- classes/circuits/circuits.h

## 7.31 qpp::make_void< Ts > Struct Template Reference

Helper for qpp::to_void<> alias template.

```
#include <traits.h>
```

### Public Types

- typedef void type

### 7.31.1 Detailed Description

**template<typename... Ts>**
**struct qpp::make_void< Ts >**

Helper for qpp::to_void<> alias template.

**See also**

qpp::to_void<>

### 7.31.2 Member Typedef Documentation

#### 7.31.2.1 type

```
template<typename... Ts>
typedef void qpp::make_void< Ts >::type
```

The documentation for this struct was generated from the following file:

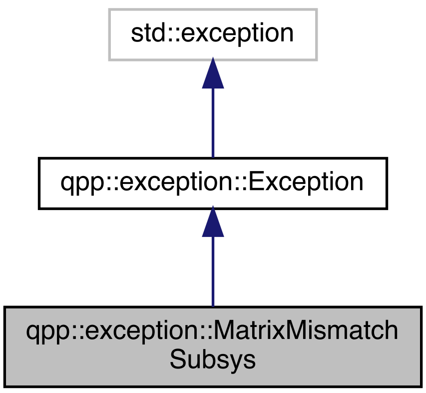
- traits.h

## 7.32 qpp::exception::MatrixMismatchSubsys Class Reference

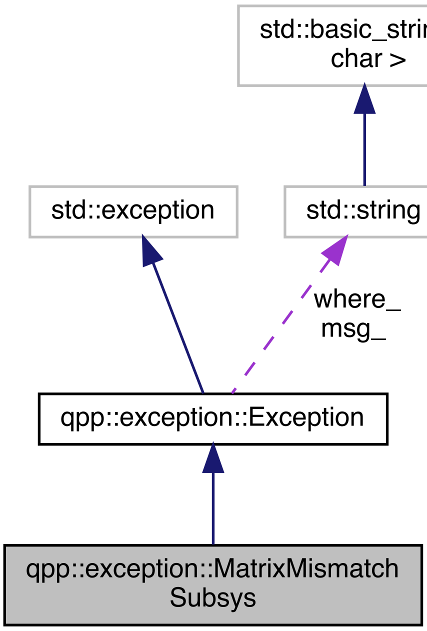
Matrix mismatch subsystems exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::MatrixMismatchSubsys:



Collaboration diagram for qpp::exception::MatrixMismatchSubsys:




## Public Member Functions

- std::string type_description () const override
  *Exception type description.*
- Exception (const std::string &where)
  *Constructs an exception.*

### 7.32.1 Detailed Description

Matrix mismatch subsystems exception.

Matrix size mismatch subsystem sizes (e.g. in qpp::apply())

### 7.32.2 Member Function Documentation

#### 7.32.2.1 Exception()

`qpp::exception::Exception::Exception [inline], [explicit]`

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.32.2.2 type_description()

```
std::string qpp::exception::MatrixMismatchSubsys::type_description ( ) const [inline], [override],
[virtual]
```

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

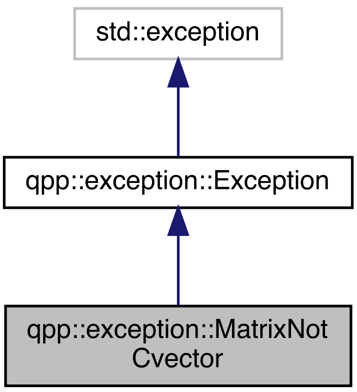
- classes/exception.h

## 7.33 qpp::exception::MatrixNotCvector Class Reference

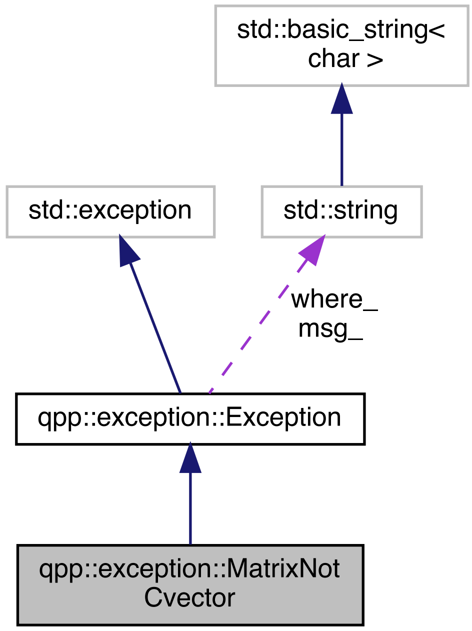
Matrix is not a column vector exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::MatrixNotCvector:



Collaboration diagram for qpp::exception::MatrixNotCvector:




## Public Member Functions

- std::string type_description () const override
    *Exception type description.*
- Exception (const std::string &where)
    *Constructs an exception.*

### 7.33.1 Detailed Description

Matrix is not a column vector exception.

Eigen::Matrix is not a column vector

### 7.33.2 Member Function Documentation

#### 7.33.2.1 Exception()

```
qpp::exception::Exception::Exception [inline], [explicit]
```

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.33.2.2 type_description()

```
std::string qpp::exception::MatrixNotCvector::type_description ( ) const  [inline], [override],
[virtual]
```

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

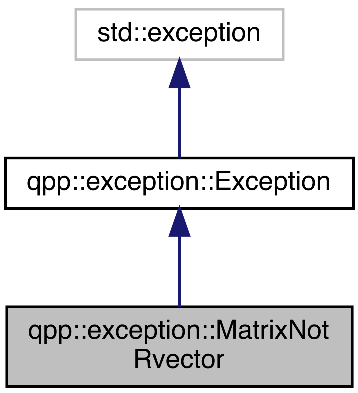
- classes/exception.h

## 7.34 qpp::exception::MatrixNotRvector Class Reference

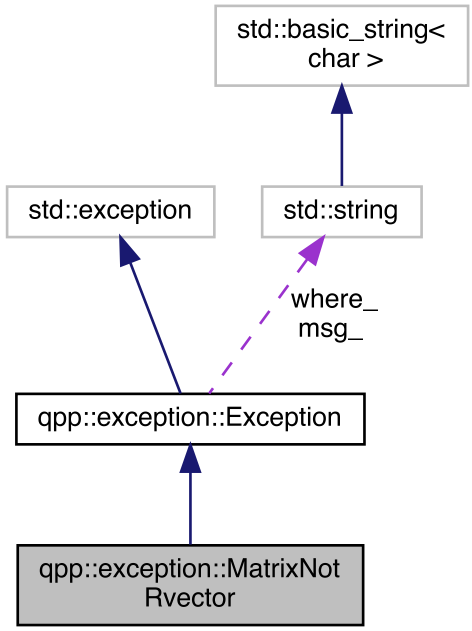
Matrix is not a row vector exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::MatrixNotRvector:



Collaboration diagram for qpp::exception::MatrixNotRvector:




## Public Member Functions

- std::string type_description () const override

    *Exception type description.*
- Exception (const std::string &where)

    *Constructs an exception.*

### 7.34.1 Detailed Description

Matrix is not a row vector exception.

Eigen::Matrix is not a row vector

### 7.34.2 Member Function Documentation

#### 7.34.2.1 Exception()

`qpp::exception::Exception::Exception [inline], [explicit]`

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.34.2.2 type_description()

```
std::string qpp::exception::MatrixNotRvector::type_description ( ) const  [inline], [override],
[virtual]
```

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

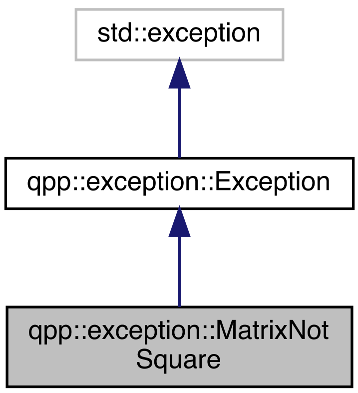
- classes/exception.h

## 7.35 qpp::exception::MatrixNotSquare Class Reference

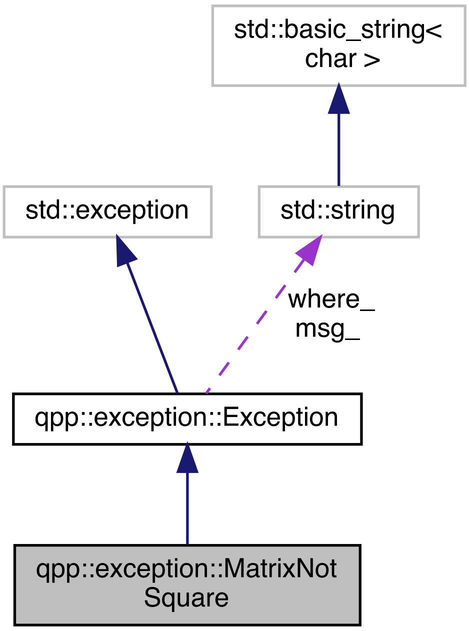
Matrix is not square exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::MatrixNotSquare:



Collaboration diagram for qpp::exception::MatrixNotSquare:




### Public Member Functions

- std::string type_description () const override

    *Exception type description.*
- Exception (const std::string &where)

    *Constructs an exception.*

## 7.35.1 Detailed Description

Matrix is not square exception.

Eigen::Matrix is not a square matrix

## 7.35.2 Member Function Documentation

### 7.35.2.1 Exception()

`qpp::exception::Exception::Exception [inline], [explicit]`

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.35.2.2 type_description()

```
std::string qpp::exception::MatrixNotSquare::type_description ( ) const  [inline], [override],
[virtual]
```

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

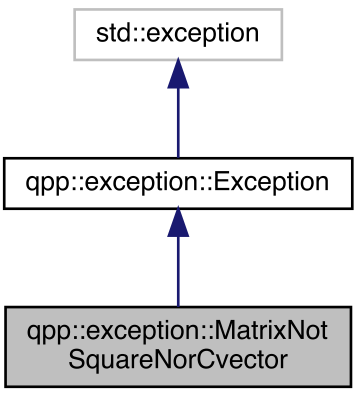
- classes/exception.h

## 7.36 qpp::exception::MatrixNotSquareNorCvector Class Reference

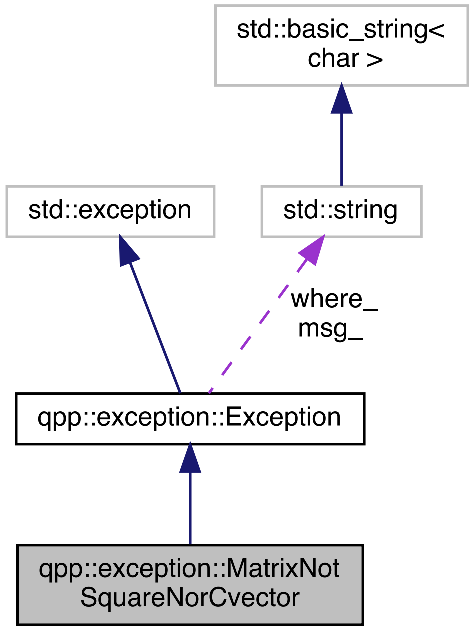
Matrix is not square nor column vector exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::MatrixNotSquareNorCvector:



Collaboration diagram for qpp::exception::MatrixNotSquareNorCvector:




## Public Member Functions

- std::string type_description () const override

  *Exception type description.*
- Exception (const std::string &where)

  *Constructs an exception.*

## 7.36.1 Detailed Description

Matrix is not square nor column vector exception.

Eigen::Matrix is not a square matrix nor a column vector

## 7.36.2 Member Function Documentation

### 7.36.2.1 Exception()

```
qpp::exception::Exception::Exception  [inline], [explicit]
```

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.36.2.2 type_description()

```
std::string qpp::exception::MatrixNotSquareNorCvector::type_description ( ) const  [inline],
[override], [virtual]
```

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

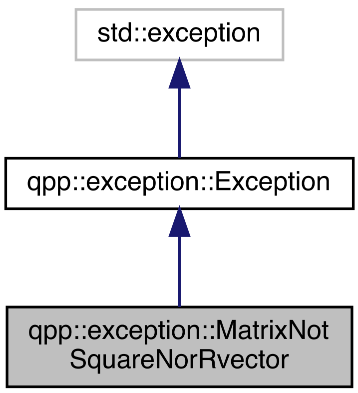
- classes/exception.h

## 7.37 qpp::exception::MatrixNotSquareNorRvector Class Reference

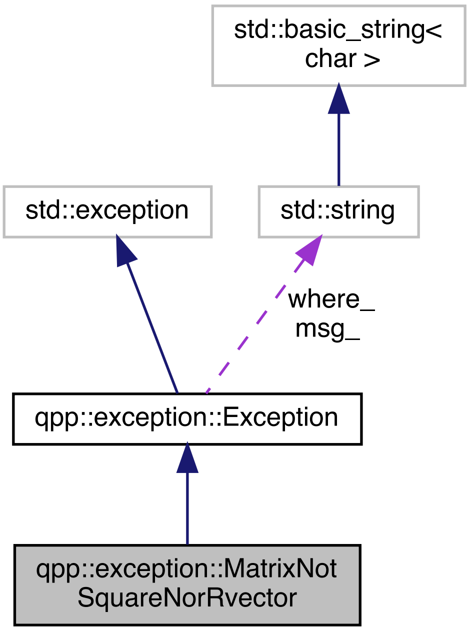
Matrix is not square nor row vector exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::MatrixNotSquareNorRvector:



Collaboration diagram for qpp::exception::MatrixNotSquareNorRvector:




### Public Member Functions

- std::string type_description () const override

    *Exception type description.*
- Exception (const std::string &where)

    *Constructs an exception.*

## 7.37.1 Detailed Description

Matrix is not square nor row vector exception.

Eigen::Matrix is not a square matrix nor a row vector

## 7.37.2 Member Function Documentation

### 7.37.2.1 Exception()

```
qpp::exception::Exception::Exception  [inline], [explicit]
```

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.37.2.2 type_description()

```
std::string qpp::exception::MatrixNotSquareNorRvector::type_description ( ) const [inline],
[override], [virtual]
```

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

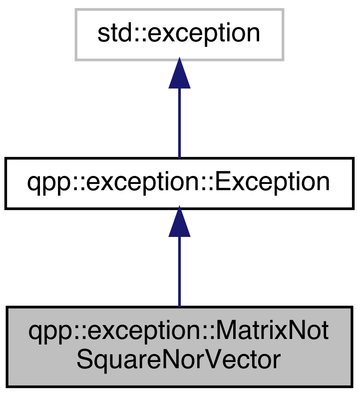
- classes/exception.h

## 7.38 qpp::exception::MatrixNotSquareNorVector Class Reference

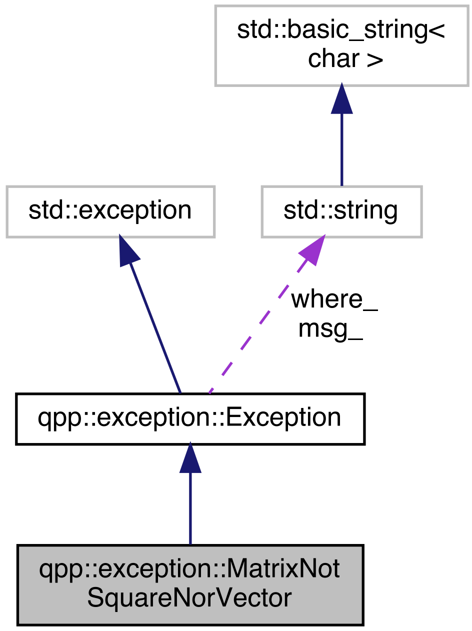
Matrix is not square nor vector exception.

`#include <classes/exception.h>`

Inheritance diagram for qpp::exception::MatrixNotSquareNorVector:



Collaboration diagram for qpp::exception::MatrixNotSquareNorVector:




### Public Member Functions

- std::string type_description () const override

    *Exception type description.*
- Exception (const std::string &where)

    *Constructs an exception.*

## 7.38.1 Detailed Description

Matrix is not square nor vector exception.

Eigen::Matrix is not a square matrix nor a row/column vector

## 7.38.2 Member Function Documentation

### 7.38.2.1 Exception()

`qpp::exception::Exception::Exception [inline], [explicit]`

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.38.2.2 type_description()

```
std::string qpp::exception::MatrixNotSquareNorVector::type_description ( ) const [inline],
[override], [virtual]
```

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

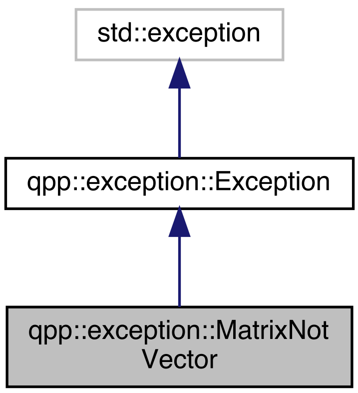
- classes/exception.h

## 7.39 qpp::exception::MatrixNotVector Class Reference

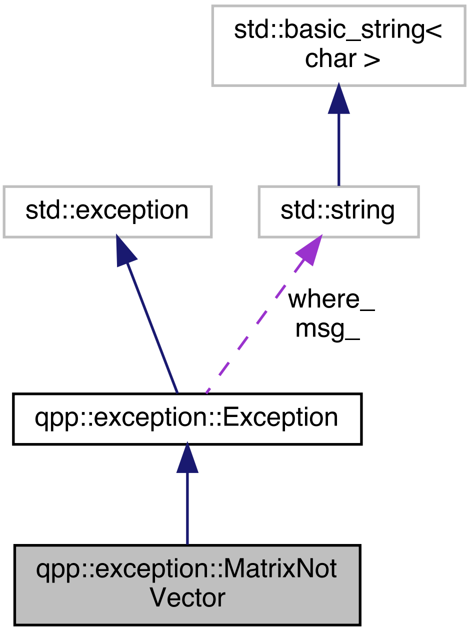
Matrix is not a vector exception.

`#include <classes/exception.h>`

Inheritance diagram for qpp::exception::MatrixNotVector:



Collaboration diagram for qpp::exception::MatrixNotVector:




## Public Member Functions

- std::string type_description () const override

  *Exception type description.*
- Exception (const std::string &where)

  *Constructs an exception.*

### 7.39.1 Detailed Description

Matrix is not a vector exception.

Eigen::Matrix is not a row or column vector

### 7.39.2 Member Function Documentation

#### 7.39.2.1 Exception()

```
qpp::exception::Exception::Exception  [inline], [explicit]
```

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.39.2.2 type_description()

```
std::string qpp::exception::MatrixNotVector::type_description ( ) const [inline], [override],
[virtual]
```

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

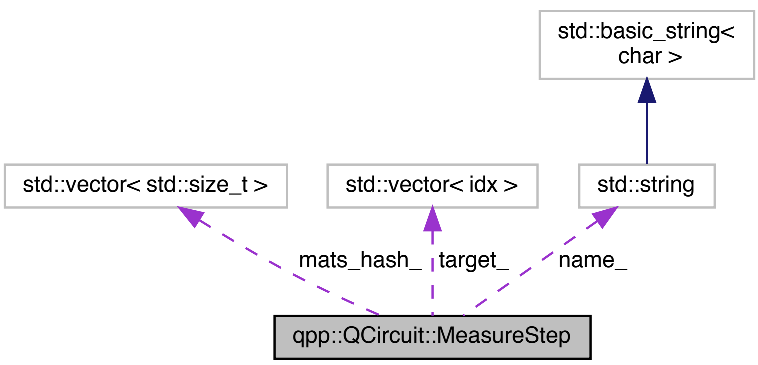
- classes/exception.h

## 7.40 qpp::QCircuit::MeasureStep Struct Reference

One step consisting only of measurements in the circuit.

```
#include <classes/circuits/circuits.h>
```

Collaboration diagram for qpp::QCircuit::MeasureStep:



## Public Member Functions

- MeasureStep ()=default

    *Default constructor.*
- MeasureStep (MeasureType measurement_type, const std::vector< std::size_t > &mats_hash, const std↩ ::vector< idx > &target, idx c_reg, std::string name={})

    *Constructs a measurement step instance.*

## Public Attributes

- MeasureType measurement_type_ = MeasureType::NONE

    *measurement type*
- std::vector< std::size_t > mats_hash_
- std::vector< idx > target_

    *target where the measurement is applied*
- idx c_reg_ {}
- std::string name_

    *custom name of the step*

### 7.40.1 Detailed Description

One step consisting only of measurements in the circuit.

### 7.40.2 Constructor & Destructor Documentation

#### 7.40.2.1 MeasureStep() [1/2]

```
qpp::QCircuit::MeasureStep::MeasureStep ( )  [default]
```

Default constructor.

#### 7.40.2.2 MeasureStep() [2/2]

```
qpp::QCircuit::MeasureStep::MeasureStep (
            MeasureType measurement_type,
            const std::vector< std::size_t > & mats_hash,
            const std::vector< idx > & target,
            idx c_reg,
            std::string name = {} )  [inline], [explicit]
```

Constructs a measurement step instance.

**Parameters**

| | |
|---|---|
| *measurement_type* | Measurement type |
| *mats_hash* | Vector of hashes of the measurement matrix/matrices |
| *target* | Target qudit indexes |
| *c_reg* | Classical register where the value of the measurement is stored |
| *name* | Optional gate name |

## 7.40.3 Member Data Documentation

### 7.40.3.1 c_reg_

```
idx qpp::QCircuit::MeasureStep::c_reg_ {}
```

index of the classical register where the measurement result is being stored

### 7.40.3.2 mats_hash_

```
std::vector<std::size_t> qpp::QCircuit::MeasureStep::mats_hash_
```

hashes of measurement matrix/matrices

### 7.40.3.3 measurement_type_

```
MeasureType qpp::QCircuit::MeasureStep::measurement_type_ = MeasureType::NONE
```

measurement type

### 7.40.3.4 name_

```
std::string qpp::QCircuit::MeasureStep::name_
```

custom name of the step

### 7.40.3.5 target_

```
std::vector<idx> qpp::QCircuit::MeasureStep::target_
```

target where the measurement is applied

The documentation for this struct was generated from the following file:

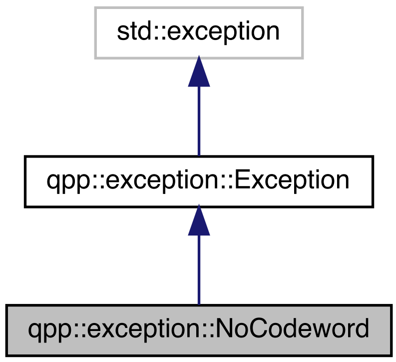
- classes/circuits/circuits.h

## 7.41 qpp::exception::NoCodeword Class Reference

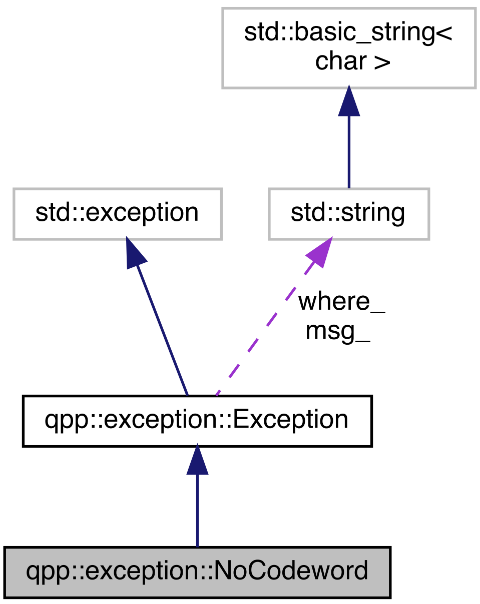
Codeword does not exist exception.

`#include <classes/exception.h>`

Inheritance diagram for qpp::exception::NoCodeword:




Collaboration diagram for qpp::exception::NoCodeword:



## Public Member Functions

- std::string type_description () const override

  *Exception type description.*
- Exception (const std::string &where)

  *Constructs an exception.*

### 7.41.1 Detailed Description

Codeword does not exist exception.

Codeword does not exist, thrown when calling qpp::Codes::codeword() with an invalid index

### 7.41.2 Member Function Documentation

#### 7.41.2.1 Exception()

```
qpp::exception::Exception::Exception  [inline], [explicit]
```

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.41.2.2 type_description()

```
std::string qpp::exception::NoCodeword::type_description ( ) const  [inline], [override],
[virtual]
```

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

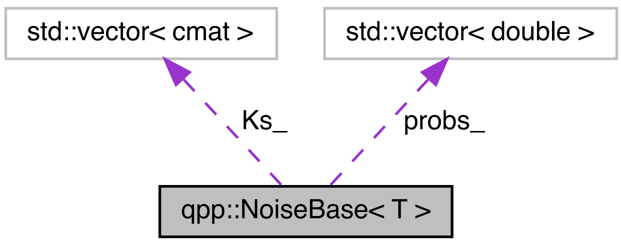
- classes/exception.h

## 7.42 qpp::NoiseBase< T > Class Template Reference

Base class for all noise models, derive your particular noise model.

```
#include <classes/noise.h>
```

Collaboration diagram for qpp::NoiseBase< T >:




### Public Types

- using noise_type = T

### Public Member Functions

- template<typename U = noise_type>
  NoiseBase (const std::vector< cmat > &Ks, typename std::enable_if< std::is_same< NoiseType::StateDependent, U >::value >::type ∗=nullptr)

  *Constructs a noise instance for StateDependent noise type.*
- template<typename U = noise_type>
  NoiseBase (const std::vector< cmat > &Ks, const std::vector< double > &probs, typename std::enable_if< std::is_same< NoiseType::StateIndependent, U >::value >::type ∗=nullptr)

  *Constructs a noise instance for StateIndependent noise type.*
- virtual ∼NoiseBase ()=default

  *Default virtual destructor.*
- idx get_d () const noexcept

  *Qudit dimension.*
- std::vector< cmat > get_Ks () const

  *Vector of noise operators.*
- std::vector< double > get_probs () const

  *Vector of probabilities corresponding to each noise operator.*
- idx get_last_idx () const

  *Index of the last occurring noise element.*
- double get_last_p () const

  *Probability of the last occurring noise element.*
- cmat get_last_K () const

  *Last occurring noise element.*

- virtual cmat operator() (const cmat &state) const

    *Function invocation operator, applies the underlying noise model on the state vector or density matrix state.*
- virtual cmat operator() (const cmat &state, idx target) const

    *Function invocation operator, applies the underlying noise model on qudit target of the multi-partite state vector or density matrix state.*
- virtual cmat operator() (const cmat &state, const std::vector< idx > &target) const

    *Function invocation operator, applies the underlying correlated noise model on qudits specified by target of the multi-partite state vector or density matrix state.*

## Protected Member Functions

- void compute_probs_ (const cmat &state, const std::vector< idx > &target) const

    *Compute probability outcomes for StateDependent noise type, otherwise returns without performing any operation (no-op)*
- cmat compute_state_ (const cmat &state, const std::vector< idx > &target) const

    *Compute the resulting state after the noise was applied.*

## Protected Attributes

- const std::vector< cmat > Ks_

    *Kraus operators.*
- std::vector< double > probs_

    *probabilities*
- idx d_ {}

    *qudit dimension*
- idx i_ {}

    *index of the last occurring noise element*
- bool generated_ {false}

    *invoked, or if the noise is state-independent*

### 7.42.1 Detailed Description

**template<class T>**
**class qpp::NoiseBase< T >**

Base class for all noise models, derive your particular noise model.

### 7.42.2 Member Typedef Documentation

#### 7.42.2.1 noise_type

```
template<class T>
using qpp::NoiseBase< T >::noise_type = T
```

### 7.42.3 Constructor & Destructor Documentation

#### 7.42.3.1 NoiseBase() [1/2]

```
template<class T>
template<typename U = noise_type>
qpp::NoiseBase< T >::NoiseBase (
            const std::vector< cmat > & Ks,
            typename std::enable_if< std::is_same< NoiseType::StateDependent, U >::value >↩
::type * = nullptr ) [inline], [explicit]
```

Constructs a noise instance for StateDependent noise type.

**Note**

SFINAEd-out for StateIndependent noise

**Parameters**

| | |
|---|---|
| *Ks* | Vector of noise (Kraus) operators that specify the noise |

#### 7.42.3.2 NoiseBase() [2/2]

```
template<class T>
template<typename U = noise_type>
qpp::NoiseBase< T >::NoiseBase (
            const std::vector< cmat > & Ks,
            const std::vector< double > & probs,
            typename std::enable_if< std::is_same< NoiseType::StateIndependent, U >::value
>::type * = nullptr ) [inline], [explicit]
```

Constructs a noise instance for StateIndependent noise type.

**Note**

SFINAEd-out for StateDependent noise

**Parameters**

| | |
|---|---|
| *Ks* | Vector of noise (Kraus) operators that specify the noise |
| *probs* | Vector of probabilities corresponding to each Kraus operator |

#### 7.42.3.3 ∼NoiseBase()

```
template<class T>
virtual qpp::NoiseBase< T >::∼NoiseBase ( )  [virtual], [default]
```

Default virtual destructor.

### 7.42.4 Member Function Documentation

#### 7.42.4.1 compute_probs_()

```
template<class T>
void qpp::NoiseBase< T >::compute_probs_ (
            const cmat & state,
            const std::vector< idx > & target ) const  [inline], [protected]
```

Compute probability outcomes for StateDependent noise type, otherwise returns without performing any operation (no-op)

**Parameters**

| | |
|---|---|
| *state* | State vector or density matrix |
| *target* | Qudit indexes where the noise is applied |

#### 7.42.4.2 compute_state_()

```
template<class T>
cmat qpp::NoiseBase< T >::compute_state_ (
            const cmat & state,
            const std::vector< idx > & target ) const  [inline], [protected]
```

Compute the resulting state after the noise was applied.

**Parameters**

| | |
|---|---|
| *state* | State vector or density matrix |
| *target* | Qudit indexes where the noise is applied |

**Returns**

Resulting state after the noise was applied

#### 7.42.4.3 get_d()

```
template<class T>
idx qpp::NoiseBase< T >::get_d ( ) const  [inline], [noexcept]
```

Qudit dimension.

**Returns**

Qudit dimension

#### 7.42.4.4 get_Ks()

```
template<class T>
std::vector<cmat> qpp::NoiseBase< T >::get_Ks ( ) const  [inline]
```

Vector of noise operators.

**Returns**

Vector of noise operators

#### 7.42.4.5 get_last_idx()

```
template<class T>
idx qpp::NoiseBase< T >::get_last_idx ( ) const  [inline]
```

Index of the last occurring noise element.

**Returns**

Index of the last occurring noise element

#### 7.42.4.6 get_last_K()

```
template<class T>
cmat qpp::NoiseBase< T >::get_last_K ( ) const  [inline]
```

Last occurring noise element.

**Returns**

Last occurring noise element

#### 7.42.4.7 get_last_p()

```
template<class T>
double qpp::NoiseBase< T >::get_last_p ( ) const  [inline]
```

Probability of the last occurring noise element.

**Returns**

Probability of the last occurring noise element

#### 7.42.4.8 get_probs()

```
template<class T>
std::vector<double> qpp::NoiseBase< T >::get_probs ( ) const  [inline]
```

Vector of probabilities corresponding to each noise operator.

**Returns**

Probability vector

#### 7.42.4.9 operator()() [1/3]

```
template<class T>
virtual cmat qpp::NoiseBase< T >::operator() (
            const cmat & state ) const  [inline], [virtual]
```

Function invocation operator, applies the underlying noise model on the state vector or density matrix *state*.

**Parameters**

| | |
|---|---|
| *state* | State vector or density matrix |

**Returns**

Resulting state vector or density matrix

#### 7.42.4.10 operator()() [2/3]

```
template<class T>
virtual cmat qpp::NoiseBase< T >::operator() (
```

```
const cmat & state,
idx target ) const [inline], [virtual]
```

Function invocation operator, applies the underlying noise model on qudit *target* of the multi-partite state vector or density matrix *state*.

**Parameters**

| | |
|---|---|
| *state* | Multi-partite state vector or density matrix |
| *target* | Qudit index where the noise is applied |

**Returns**

Resulting state vector or density matrix

#### 7.42.4.11 operator()() [3/3]

```
template<class T>
virtual cmat qpp::NoiseBase< T >::operator() (
            const cmat & state,
            const std::vector< idx > & target ) const [inline], [virtual]
```

Function invocation operator, applies the underlying correlated noise model on qudits specified by *target* of the multi-partite state vector or density matrix *state*.

**Parameters**

| | |
|---|---|
| *state* | Multi-partite state vector or density matrix |
| *target* | Qudit indexes where the correlated noise is applied |

**Returns**

Resulting state vector or density matrix

### 7.42.5 Member Data Documentation

#### 7.42.5.1 d_

```
template<class T>
idx qpp::NoiseBase< T >::d_ {} [mutable], [protected]
```

qudit dimension

#### 7.42.5.2 generated_

```
template<class T>
bool qpp::NoiseBase< T >::generated_ {false} [mutable], [protected]
```

invoked, or if the noise is state-independent

set to true after compute_state_() is

#### 7.42.5.3 i_

```
template<class T>
idx qpp::NoiseBase< T >::i_ {} [mutable], [protected]
```

index of the last occurring noise element

#### 7.42.5.4 Ks_

```
template<class T>
const std::vector<cmat> qpp::NoiseBase< T >::Ks_ [protected]
```

Kraus operators.

#### 7.42.5.5 probs_

```
template<class T>
std::vector<double> qpp::NoiseBase< T >::probs_ [mutable], [protected]
```

probabilities

The documentation for this class was generated from the following file:

- classes/noise.h

## 7.43 qpp::NoiseType Class Reference

Contains template tags used to specify the noise type.

`#include <classes/noise.h>`

### Classes

- class StateDependent

  *Template tag, used whenever the noise is state-dependent.*
- class StateIndependent

  *Template tag, used whenever the noise is state-independent.*

### 7.43.1 Detailed Description

Contains template tags used to specify the noise type.

The documentation for this class was generated from the following file:

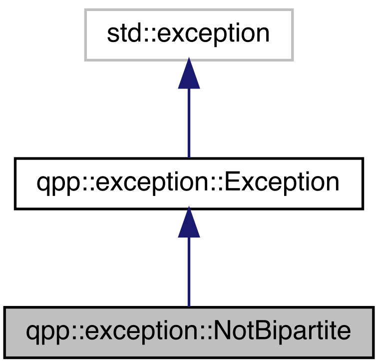
- classes/noise.h

## 7.44 qpp::exception::NotBipartite Class Reference

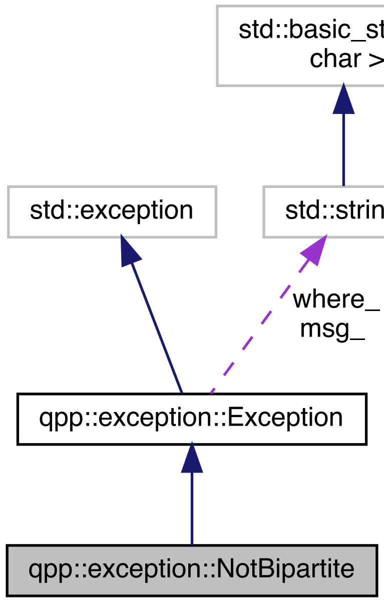
Not bi-partite exception.

`#include <classes/exception.h>`

Inheritance diagram for qpp::exception::NotBipartite:



Collaboration diagram for qpp::exception::NotBipartite:




### Public Member Functions

- std::string type_description () const override

    *Exception type description.*
- Exception (const std::string &where)

    *Constructs an exception.*

## 7.44.1 Detailed Description

Not bi-partite exception.

std::vector<idx> of dimensions has size different from 2

## 7.44.2 Member Function Documentation

### 7.44.2.1 Exception()

`qpp::exception::Exception::Exception [inline], [explicit]`

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.44.2.2 type_description()

```
std::string qpp::exception::NotBipartite::type_description ( ) const  [inline], [override],
[virtual]
```

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

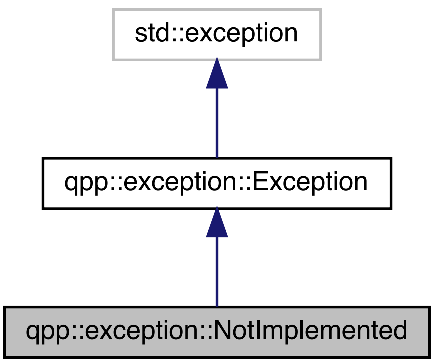
- classes/exception.h

## 7.45 qpp::exception::NotImplemented Class Reference

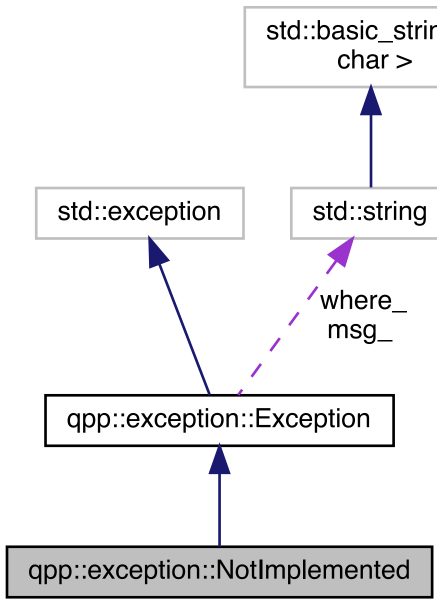
Code not yet implemented.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::NotImplemented:



Collaboration diagram for qpp::exception::NotImplemented:




## Public Member Functions

- std::string type_description () const override

    *Exception type description.*
- Exception (const std::string &where)

    *Constructs an exception.*

### 7.45.1 Detailed Description

Code not yet implemented.

### 7.45.2 Member Function Documentation

#### 7.45.2.1 Exception()

`qpp::exception::Exception::Exception  [inline], [explicit]`

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.45.2.2 type_description()

```
std::string qpp::exception::NotImplemented::type_description ( ) const  [inline], [override],
[virtual]
```

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

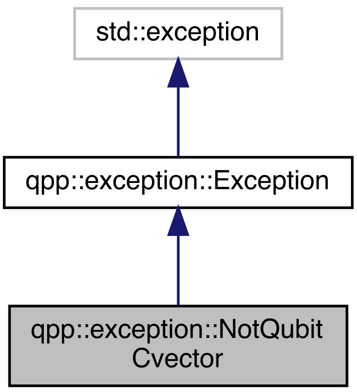
- classes/exception.h

## 7.46 qpp::exception::NotQubitCvector Class Reference

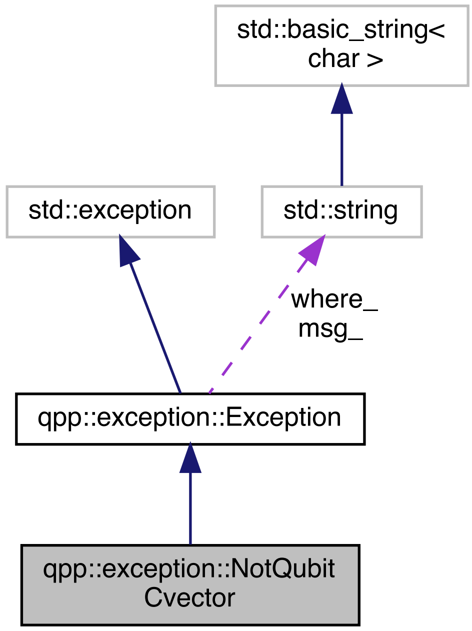
Column vector is not 2 x 1 exception.

`#include <classes/exception.h>`

Inheritance diagram for qpp::exception::NotQubitCvector:



Collaboration diagram for qpp::exception::NotQubitCvector:




## Public Member Functions

- std::string type_description () const override

    *Exception type description.*
- Exception (const std::string &where)

    *Constructs an exception.*

### 7.46.1 Detailed Description

Column vector is not 2 x 1 exception.

Eigen::Matrix is not 2 x 1

### 7.46.2 Member Function Documentation

#### 7.46.2.1 Exception()

```
qpp::exception::Exception::Exception  [inline], [explicit]
```

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.46.2.2 type_description()

`std::string qpp::exception::NotQubitCvector::type_description ( ) const [inline], [override], [virtual]`

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

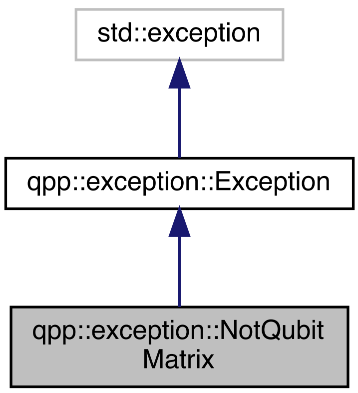
- classes/exception.h

## 7.47 qpp::exception::NotQubitMatrix Class Reference

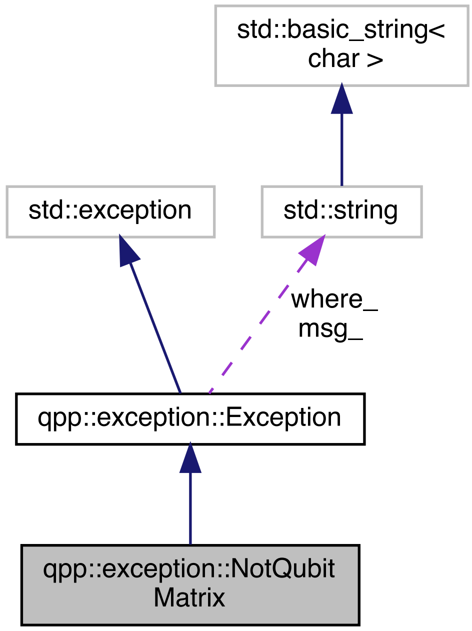
Matrix is not 2 x 2 exception.

`#include <classes/exception.h>`

Inheritance diagram for qpp::exception::NotQubitMatrix:



Collaboration diagram for qpp::exception::NotQubitMatrix:




### Public Member Functions

- std::string type_description () const override

    *Exception type description.*
- Exception (const std::string &where)

    *Constructs an exception.*

## 7.47.1 Detailed Description

Matrix is not 2 x 2 exception.

Eigen::Matrix is not 2 x 2

## 7.47.2 Member Function Documentation

### 7.47.2.1 Exception()

`qpp::exception::Exception::Exception  [inline], [explicit]`

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.47.2.2 type_description()

```
std::string qpp::exception::NotQubitMatrix::type_description ( ) const  [inline], [override],
[virtual]
```

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

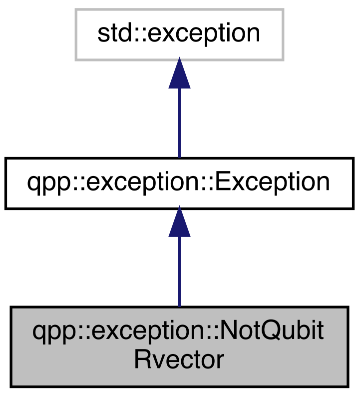
- classes/exception.h

## 7.48 qpp::exception::NotQubitRvector Class Reference

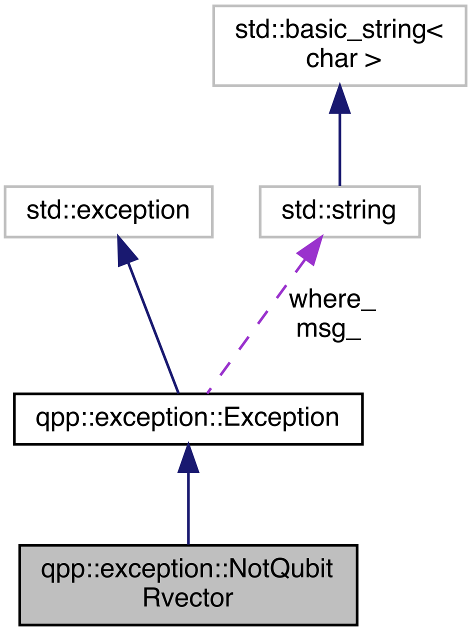
Row vector is not 1 x 2 exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::NotQubitRvector:



Collaboration diagram for qpp::exception::NotQubitRvector:




## Public Member Functions

- std::string type_description () const override

    *Exception type description.*
- Exception (const std::string &where)

    *Constructs an exception.*

### 7.48.1 Detailed Description

Row vector is not 1 x 2 exception.

Eigen::Matrix is not 1 x 2

### 7.48.2 Member Function Documentation

#### 7.48.2.1 Exception()

```
qpp::exception::Exception::Exception [inline], [explicit]
```

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.48.2.2 type_description()

`std::string qpp::exception::NotQubitRvector::type_description ( ) const [inline], [override], [virtual]`

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

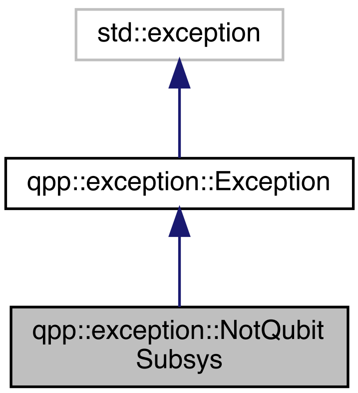
- classes/exception.h

## 7.49 qpp::exception::NotQubitSubsys Class Reference

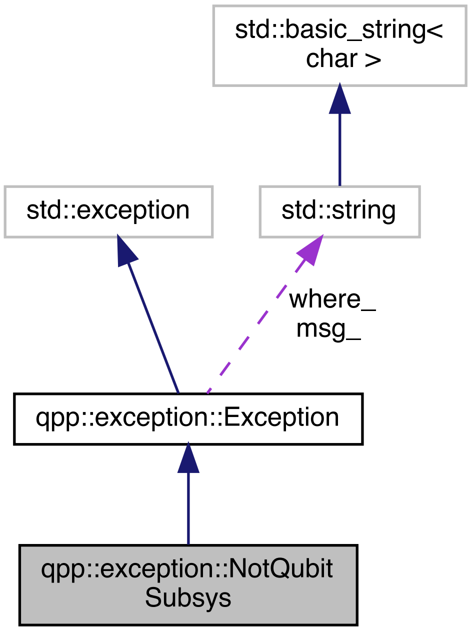
Subsystems are not qubits exception.

`#include <classes/exception.h>`

Inheritance diagram for qpp::exception::NotQubitSubsys:



Collaboration diagram for qpp::exception::NotQubitSubsys:




## Public Member Functions

- std::string type_description () const override
  *Exception type description.*
- Exception (const std::string &where)
  *Constructs an exception.*

### 7.49.1 Detailed Description

Subsystems are not qubits exception.

Subsystems are not 2-dimensional (qubits)

### 7.49.2 Member Function Documentation

#### 7.49.2.1 Exception()

```
qpp::exception::Exception::Exception  [inline], [explicit]
```

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.49.2.2 type_description()

```
std::string qpp::exception::NotQubitSubsys::type_description ( ) const [inline], [override],
[virtual]
```

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

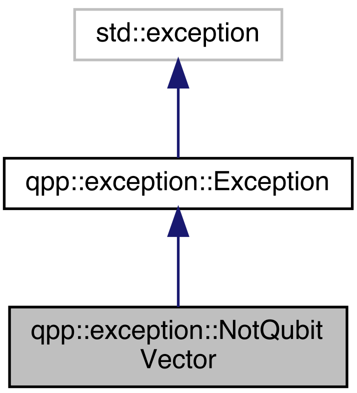
- classes/exception.h

## 7.50 qpp::exception::NotQubitVector Class Reference

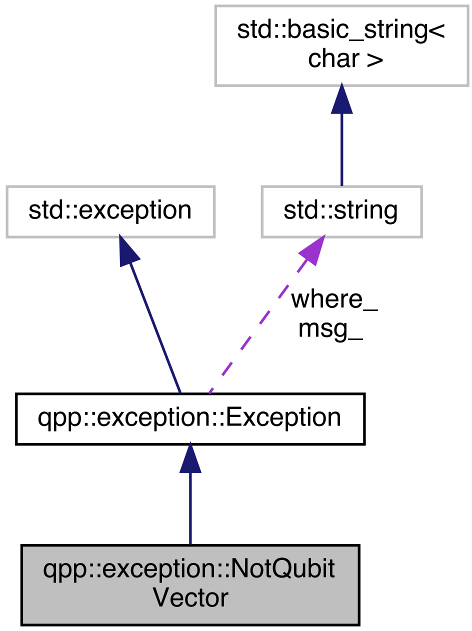
Vector is not 2 x 1 nor 1 x 2 exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::NotQubitVector:



Collaboration diagram for qpp::exception::NotQubitVector:




## Public Member Functions

- std::string type_description () const override

    *Exception type description.*
- Exception (const std::string &where)

    *Constructs an exception.*

### 7.50.1 Detailed Description

Vector is not 2 x 1 nor 1 x 2 exception.

Eigen::Matrix is not 2 x 1 nor 1 x 2

### 7.50.2 Member Function Documentation

#### 7.50.2.1 Exception()

```
qpp::exception::Exception::Exception  [inline], [explicit]
```

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.50.2.2 type_description()

```
std::string qpp::exception::NotQubitVector::type_description ( ) const  [inline], [override],
[virtual]
```

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

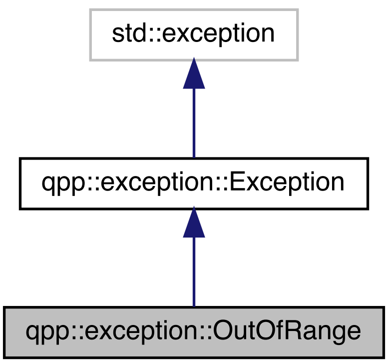
- classes/exception.h

## 7.51 qpp::exception::OutOfRange Class Reference

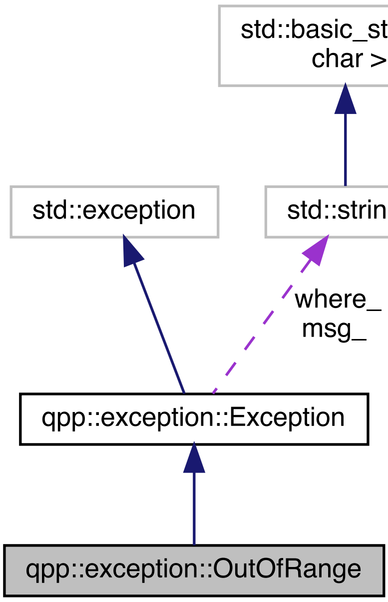
Argument out of range exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::OutOfRange:



Collaboration diagram for qpp::exception::OutOfRange:




## Public Member Functions

- std::string type_description () const override

    *Exception type description.*
- Exception (const std::string &where)

    *Constructs an exception.*

### 7.51.1 Detailed Description

Argument out of range exception.

Argument out of range

### 7.51.2 Member Function Documentation

#### 7.51.2.1 Exception()

```
qpp::exception::Exception::Exception  [inline], [explicit]
```

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.51.2.2 type_description()

```
std::string qpp::exception::OutOfRange::type_description ( ) const  [inline], [override],
[virtual]
```

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

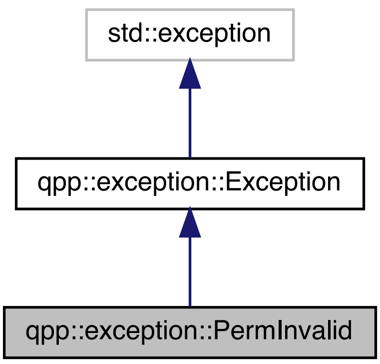
- classes/exception.h

## 7.52 qpp::exception::PermInvalid Class Reference

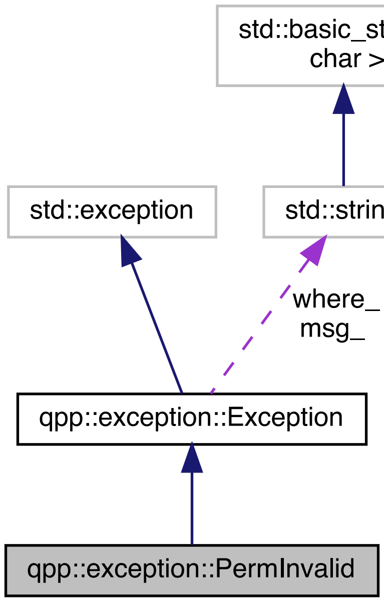
Invalid permutation exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::PermInvalid:



Collaboration diagram for qpp::exception::PermInvalid:




### Public Member Functions

- std::string type_description () const override

  *Exception type description.*
- Exception (const std::string &where)

  *Constructs an exception.*

## 7.52.1 Detailed Description

Invalid permutation exception.

std::vector<idx> does note represent a valid permutation

## 7.52.2 Member Function Documentation

### 7.52.2.1 Exception()

`qpp::exception::Exception::Exception [inline], [explicit]`

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.52.2.2 type_description()

```
std::string qpp::exception::PermInvalid::type_description ( ) const [inline], [override],
[virtual]
```

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

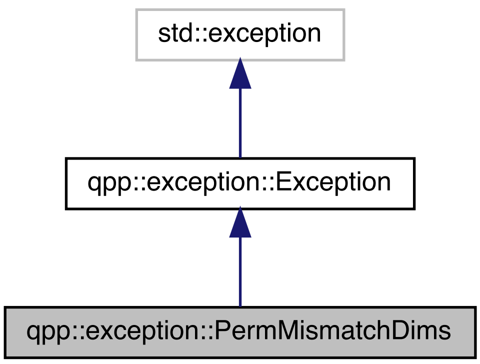
- classes/exception.h

## 7.53 qpp::exception::PermMismatchDims Class Reference

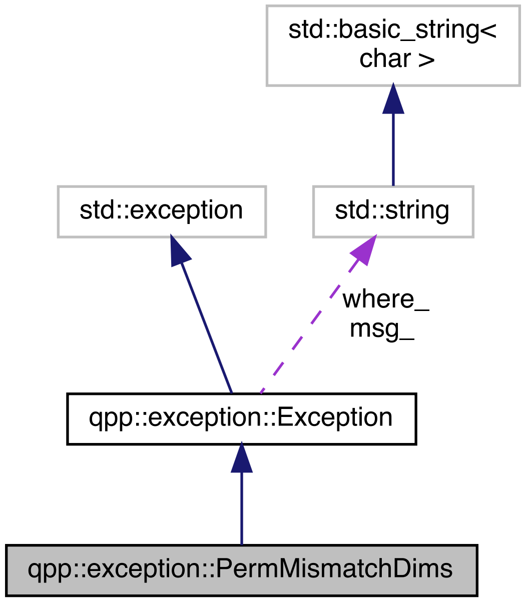
Permutation mismatch dimensions exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::PermMismatchDims:



Collaboration diagram for qpp::exception::PermMismatchDims:




### Public Member Functions

- std::string type_description () const override

    *Exception type description.*
- Exception (const std::string &where)

    *Constructs an exception.*

## 7.53.1 Detailed Description

Permutation mismatch dimensions exception.

Size of the std::vector<idx> representing the permutation is different from the size of the std::vector<idx> of dimensions

## 7.53.2 Member Function Documentation

### 7.53.2.1 Exception()

```
qpp::exception::Exception::Exception  [inline], [explicit]
```

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.53.2.2 type_description()

```
std::string qpp::exception::PermMismatchDims::type_description ( ) const [inline], [override],
[virtual]
```

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

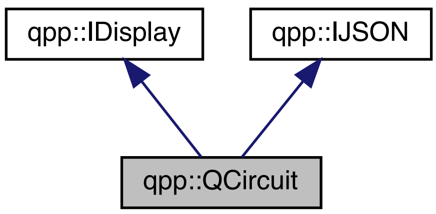
- classes/exception.h

## 7.54 qpp::QCircuit Class Reference

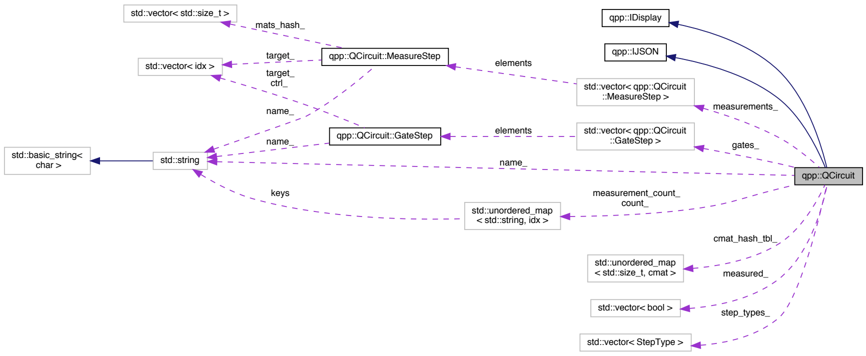
Quantum circuit class.

```
#include <classes/circuits/circuits.h>
```

Inheritance diagram for qpp::QCircuit:



Collaboration diagram for qpp::QCircuit:




## Classes

- struct GateStep

    *One step consisting only of gates/operators in the circuit.*
- class iterator

    *Quantum circuit bound-checking (safe) iterator.*
- struct MeasureStep

    *One step consisting only of measurements in the circuit.*

## Public Types

- enum GateType {
  GateType::NONE, GateType::SINGLE, GateType::TWO, GateType::THREE,
  GateType::CUSTOM, GateType::FAN, GateType::SINGLE_CTRL_SINGLE_TARGET, GateType::SINGLE_CTRL_MULTIPLE_
  GateType::MULTIPLE_CTRL_SINGLE_TARGET, GateType::MULTIPLE_CTRL_MULTIPLE_TARGET,
  GateType::CUSTOM_CTRL, GateType::SINGLE_cCTRL_SINGLE_TARGET,
  GateType::SINGLE_cCTRL_MULTIPLE_TARGET, GateType::MULTIPLE_cCTRL_SINGLE_TARGET,
  GateType::MULTIPLE_cCTRL_MULTIPLE_TARGET, GateType::CUSTOM_cCTRL }

    *Type of gate being executed in a gate step.*
- enum MeasureType { MeasureType::NONE, MeasureType::MEASURE_Z, MeasureType::MEASURE_V, MeasureType::MEASURE_V_MANY }

    *Type of measurement being executed in a measurement step.*
- enum StepType { StepType::NONE, StepType::GATE, StepType::MEASUREMENT, StepType::NOP }

    *Types of each step in the quantum circuit.*
- using const_iterator = iterator

    *both iterators are const_iterators*

## Public Member Functions

- iterator begin ()

    *Iterator to the first element.*
- const_iterator begin () const noexcept

    *Constant iterator to the first element.*
- const_iterator cbegin () const noexcept

*Constant iterator to the first element.*
- iterator end ()

  *Iterator to the next to the last element.*
- const_iterator end () const noexcept

  *Constant iterator to the next to the last element.*
- const_iterator cend () const noexcept

  *Constant iterator to the next to the last element.*
- QCircuit (idx nq, idx nc=0, idx d=2, std::string name={})

  *Constructs a quantum circuit.*
- virtual ~QCircuit ()=default

  *Default virtual destructor.*
- idx get_nq () const noexcept

  *Total number of qudits in the circuit.*
- idx get_nc () const noexcept

  *Total number of classical dits in the circuit.*
- idx get_d () const noexcept

  *Dimension of the comprising qudits.*
- std::string get_name () const

  *Quantum circuit name.*
- idx get_measured (idx i) const

  *Check whether qudit i was already measured.*
- std::vector< idx > get_measured () const

  *Vector of already measured qudit indexes.*
- std::vector< idx > get_non_measured () const

  *Vector of non-measured qudit indexes.*
- idx get_gate_count (const std::string &name={}) const

  *Quantum circuit gate count.*
- idx get_gate_depth (const std::string &name={}) const

  *Quantum circuit gate depth.*
- idx get_measurement_count () const noexcept

  *Quantum circuit total measurement count.*
- idx get_measurement_count (const std::string &name) const

  *Quantum circuit measurement count.*
- idx get_step_count () const noexcept

  *Quantum circuit total steps count, i.e. the sum of gate count and measurement count.*
- idx get_nop_count () const

  *No-op count.*
- QCircuit & gate (const cmat &U, idx i, std::string name={})

  *Applies the single qudit gate U on single qudit i.*
- QCircuit & gate (const cmat &U, idx i, idx j, std::string name={})

  *Applies the two qudit gate U on qudits i and j.*
- QCircuit & gate (const cmat &U, idx i, idx j, idx k, std::string name={})

  *Applies the three qudit gate U on qudits i, j and k.*
- QCircuit & gate_fan (const cmat &U, const std::vector< idx > &target, std::string name={})

  *Applies the single qudit gate U on every qudit listed in target.*
- QCircuit & gate_fan (const cmat &U, const std::initializer_list< idx > &target, std::string name={})

  *Applies the single qudit gate U on every qudit listed in target.*
- QCircuit & gate_fan (const cmat &U, std::string name={})

  *Applies the single qudit gate U on every remaining non-measured qudit.*
- QCircuit & gate_custom (const cmat &U, const std::vector< idx > &target, std::string name={})

  *Jointly applies the custom multiple qudit gate U on the qudit indexes specified by target.*

- QCircuit & QFT (const std::vector< idx > &target, bool swap=true)

  *Applies the quantum Fourier transform (as a series of gates) on the qudit indexes specified by target.*
- QCircuit & QFT (const std::initializer_list< idx > &target, bool swap=true)

  *Applies the quantum Fourier transform (as a series of gates) on the qudit indexes specified by target.*
- QCircuit & QFT (bool swap=true)

  *Applies the quantum Fourier transform (as a series of gates) on all of remaining non-measured qudits.*
- QCircuit & TFQ (const std::vector< idx > &target, bool swap QPP_UNUSED_=true)

  *Applies the inverse quantum Fourier transform (as a series of gates) on the qudit indexes specified by target.*
- QCircuit & TFQ (const std::initializer_list< idx > &target, bool swap=true)

  *Applies the inverse quantum Fourier transform (as a series of gates) on the qudit indexes specified by target.*
- QCircuit & TFQ (bool swap=true)

  *Applies the inverse quantum Fourier transform (as a series of gates) on all of remaining non-measured qudits.*
- QCircuit & CTRL (const cmat &U, idx ctrl, idx target, std::string name={})

  *Applies the single qudit controlled gate U with control qudit ctrl and target qudit target.*
- QCircuit & CTRL (const cmat &U, idx ctrl, const std::vector< idx > &target, std::string name={})

  *Applies the single qudit controlled gate U with control qudit ctrl on every qudit listed in target.*
- QCircuit & CTRL (const cmat &U, const std::vector< idx > &ctrl, idx target, std::string name={})

  *Applies the single qudit controlled gate U with multiple control qudits listed in ctrl on the target qudit target.*
- QCircuit & CTRL (const cmat &U, const std::vector< idx > &ctrl, const std::vector< idx > &target, std::string name={})

  *Applies the single qudit controlled gate U with multiple control qudits listed in ctrl on every qudit listed in target.*
- QCircuit & CTRL_custom (const cmat &U, const std::vector< idx > &ctrl, const std::vector< idx > &target, std::string name={})

  *Jointly applies the custom multiple-qudit controlled gate U with multiple control qudits listed in ctrl on the qudit indexes specified by target.*
- QCircuit & cCTRL (const cmat &U, idx ctrl_dit, idx target, std::string name={})

  *Applies the single qubit controlled gate U with classical control dit ctrl and target qudit target.*
- QCircuit & cCTRL (const cmat &U, idx ctrl_dit, const std::vector< idx > &target, std::string name={})

  *Applies the single qudit controlled gate U with classical control dit ctrl on every qudit listed in target.*
- QCircuit & cCTRL (const cmat &U, const std::vector< idx > &ctrl_dits, idx target, std::string name={})

  *Applies the single qudit controlled gate U with multiple classical control dits listed in ctrl on the target qudit target.*
- QCircuit & cCTRL (const cmat &U, const std::vector< idx > &ctrl_dits, const std::vector< idx > &target, std::string name={})

  *Applies the single qudit controlled gate U with multiple classical control dits listed in ctrl on every qudit listed in target.*
- QCircuit & cCTRL_custom (const cmat &U, const std::vector< idx > &ctrl_dits, const std::vector< idx > &target, std::string name={})

  *Jointly applies the custom multiple-qudit controlled gate U with multiple classical control dits listed in ctrl on the qudit indexes specified by target.*
- QCircuit & measureZ (idx target, idx c_reg, std::string name={})

  *Measurement of single qudit in the computational basis (Z-basis)*
- QCircuit & measureV (const cmat &V, idx target, idx c_reg, std::string name={})

  *Measurement of single qudit in the orthonormal basis or rank-1 projectors specified by the columns of matrix V.*
- QCircuit & measureV (const cmat &V, const std::vector< idx > &target, idx c_reg, std::string name={})

  *Joint measurement of multiple qudits in the orthonormal basis or rank-1 projectors specified by the columns of matrix V.*
- QCircuit & nop ()

  *No operation (no-op)*
- std::string to_JSON (bool enclosed_in_curly_brackets=true) const override

  *qpp::IJOSN::to_JSON() override*

## Private Member Functions

- void add_hash_ (const cmat &U, std::size_t hashU)

  *Adds matrix to the hash table.*
- const std::vector< MeasureStep > & get_measurements_ () const noexcept

  *Vector of qpp::QCircuit::MeasureStep.*
- const std::vector< GateStep > & get_gates_ () const noexcept

  *Vector of qpp::QCircuit::GateStep.*
- const std::unordered_map< std::size_t, cmat > & get_cmat_hash_tbl_ () const noexcept

  *Hash table with the matrices used in the circuit.*
- std::ostream & display (std::ostream &os) const override

  *qpp::IDisplay::display() override*

## Private Attributes

- const idx nq_

  *number of qudits*
- const idx nc_

  *number of classical "dits"*
- const idx d_

  *qudit dimension*
- std::string name_

  *optional circuit name*
- std::vector< bool > measured_

  *keeps track of the measured qudits*
- std::unordered_map< std::size_t, cmat > cmat_hash_tbl_ {}
- std::unordered_map< std::string, idx > count_ {}

  *gate counts*
- std::unordered_map< std::string, idx > measurement_count_ {}

  *measurement counts*
- std::vector< GateStep > gates_ {}

  *gates*
- std::vector< MeasureStep > measurements_ {}

  *measurements*
- std::vector< StepType > step_types_ {}

  *type of each step*

## Friends

- class QEngine
- std::ostream & operator<< (std::ostream &os, const GateType &gate_type)

  *Extraction operator overload for qpp::QCircuit::GateType enum class.*
- std::ostream & operator<< (std::ostream &os, const GateStep &gate_step)

  *Extraction operator overload for qpp::QCircuit::GateStep class.*
- std::ostream & operator<< (std::ostream &os, const MeasureType &measure_type)

  *Extraction operator overload for qpp::QCircuit::MeasureType enum class.*
- std::ostream & operator<< (std::ostream &os, const MeasureStep &measure_step)

  *Extraction operator overload for qpp::QCircuit::MeasureStep class.*

### 7.54.1 Detailed Description

Quantum circuit class.

**See also**

qpp::QEngine

### 7.54.2 Member Typedef Documentation

#### 7.54.2.1 const_iterator

```
using qpp::QCircuit::const_iterator = iterator
```

both iterators are const_iterators

### 7.54.3 Member Enumeration Documentation

#### 7.54.3.1 GateType

```
enum qpp::QCircuit::GateType  [strong]
```

Type of gate being executed in a gate step.

**Enumerator**

| | |
|---:|---|
| NONE | represents no gate |
| SINGLE | unitary gate on a single qudit |
| TWO | unitary gate on 2 qudits |
| THREE | unitary gate on 3 qudits |
| CUSTOM | custom gate on multiple qudits |
| FAN | same unitary gate on multiple qudits |
| SINGLE_CTRL_SINGLE_TARGET | controlled 1 qudit unitary gate with one control and one target |
| SINGLE_CTRL_MULTIPLE_TARGET | controlled 1 qudit unitary gate with one control and multiple targets |
| MULTIPLE_CTRL_SINGLE_TARGET | controlled 1 qudit unitary gate with multiple controls and single target |
| MULTIPLE_CTRL_MULTIPLE_TARGET | controlled 1 qudit unitary gate with multiple controls and multiple targets |
| CUSTOM_CTRL | custom controlled gate with multiple controls and multiple targets |
| SINGLE_cCTRL_SINGLE_TARGET | controlled 1 qudit unitary gate with one classical control and one target |
| SINGLE_cCTRL_MULTIPLE_TARGET | controlled 1 qudit unitary gate with one classical control and multiple targets |
| MULTIPLE_cCTRL_SINGLE_TARGET | controlled 1 qudit unitary gate with multiple classical controls and single target |
| MULTIPLE_cCTRL_MULTIPLE_TARGET | controlled 1 qudit unitary gate with multiple classical controls and multiple targets |
| CUSTOM_cCTRL | custom controlled gate with multiple controls and multiple targets |

#### 7.54.3.2 MeasureType

```
enum qpp::QCircuit::MeasureType [strong]
```

Type of measurement being executed in a measurement step.

**Enumerator**

| | |
|---|---|
| NONE | represents no measurement |
| MEASURE_Z | Z measurement of single qudit. |
| MEASURE_V | measurement of single qudit in the orthonormal basis or rank-1 projectors specified by the columns of matrix *V* |
| MEASURE_V_MANY | measurement of multiple qudits in the orthonormal basis or rank-1 projectors specified by the columns of matrix *V* |

#### 7.54.3.3 StepType

```
enum qpp::QCircuit::StepType [strong]
```

Types of each step in the quantum circuit.

**Enumerator**

| | |
|---|---|
| NONE | represents no step |
| GATE | quantum gate(s) |
| MEASUREMENT | measurement |
| NOP | no-op |

### 7.54.4 Constructor & Destructor Documentation

#### 7.54.4.1 QCircuit()

```
qpp::QCircuit::QCircuit (
            idx nq,
            idx nc = 0,
            idx d = 2,
            std::string name = {} )  [inline], [explicit]
```

Constructs a quantum circuit.

**Note**

The measurement results can only be stored in the classical dits of which number is specified by *nc*

**Parameters**

| | |
|---|---|
| *nq* | Number of qbits |
| *nc* | Number of classical dits (optional) |
| *d* | Subsystem dimensions (optional, default is qubit, i.e. $d = 2$) |
| *name* | Circuit name (optional) |

#### 7.54.4.2 ∼QCircuit()

```
virtual qpp::QCircuit::∼QCircuit ( )  [virtual], [default]
```

Default virtual destructor.

### 7.54.5 Member Function Documentation

#### 7.54.5.1 add_hash_()

```
void qpp::QCircuit::add_hash_ (
            const cmat & U,
            std::size_t hashU )  [inline], [private]
```

Adds matrix to the hash table.

**Note**

Throws if a hash collision is detected., i.e., if two different matrices have the same hash

**Parameters**

| | |
|---|---|
| *U* | Complex matrix |
| *hashU* | Hash value of U |

#### 7.54.5.2 begin() [1/2]

```
iterator qpp::QCircuit::begin ( )  [inline]
```

Iterator to the first element.

**Returns**

Iterator to the first element

#### 7.54.5.3 begin() [2/2]

```
const_iterator qpp::QCircuit::begin ( ) const  [inline], [noexcept]
```

Constant iterator to the first element.

**Returns**

Constant iterator to the first element

#### 7.54.5.4 cbegin()

```
const_iterator qpp::QCircuit::cbegin ( ) const  [inline], [noexcept]
```

Constant iterator to the first element.

**Returns**

Constant iterator to the first element

#### 7.54.5.5 cCTRL() [1/4]

```
QCircuit& qpp::QCircuit::cCTRL (
             const cmat & U,
             idx ctrl_dit,
             idx target,
             std::string name = {} )  [inline]
```

Applies the single qubit controlled gate *U* with classical control dit *ctrl* and target qudit *target*.

**Parameters**

| | |
|---|---|
| *U* | Single qudit quantum gate |
| *ctrl_dit* | Classical control dit index |
| *target* | Target qudit index |
| *name* | Optional gate name |

**Returns**

Reference to the current instance

#### 7.54.5.6 cCTRL() [2/4]

```
QCircuit& qpp::QCircuit::cCTRL (
            const cmat & U,
            idx ctrl_dit,
            const std::vector< idx > & target,
            std::string name = {} ) [inline]
```

Applies the single qudit controlled gate *U* with classical control dit *ctrl* on every qudit listed in *target*.

**Parameters**

| | |
|---|---|
| *U* | Single qudit quantum gate |
| *ctrl_dit* | Classical control dit index |
| *target* | Target qudit indexes; the gate *U* is applied on every one of them depending on the values of the classical control dits |
| *name* | Optional gate name |

**Returns**

Reference to the current instance

#### 7.54.5.7 cCTRL() [3/4]

```
QCircuit& qpp::QCircuit::cCTRL (
            const cmat & U,
            const std::vector< idx > & ctrl_dits,
            idx target,
            std::string name = {} ) [inline]
```

Applies the single qudit controlled gate *U* with multiple classical control dits listed in *ctrl* on the target qudit *target*.

**Parameters**

| | |
|---|---|
| *U* | Single qudit quantum gate |
| *ctrl_dits* | Classical control dits indexes |
| *target* | Target qudit index |
| *name* | Optional gate name |

**Returns**

Reference to the current instance

#### 7.54.5.8 cCTRL() [4/4]

```
QCircuit& qpp::QCircuit::cCTRL (
            const cmat & U,
```

```
const std::vector< idx > & ctrl_dits,
const std::vector< idx > & target,
std::string name = {} ) [inline]
```

Applies the single qudit controlled gate *U* with multiple classical control dits listed in *ctrl* on every qudit listed in *target*.

**Parameters**

| | |
|---|---|
| *U* | Single qudit quantum gate |
| *ctrl_dits* | Classical control dits indexes |
| *target* | Target qudit indexes; the gate *U* is applied on every one of them depending on the values of the classical control dits |
| *name* | Optional gate name |

**Returns**

Reference to the current instance

#### 7.54.5.9 cCTRL_custom()

```
QCircuit& qpp::QCircuit::cCTRL_custom (
        const cmat & U,
        const std::vector< idx > & ctrl_dits,
        const std::vector< idx > & target,
        std::string name = {} ) [inline]
```

Jointly applies the custom multiple-qudit controlled gate *U* with multiple classical control dits listed in *ctrl* on the qudit indexes specified by *target*.

**Parameters**

| | |
|---|---|
| *U* | Multiple-qudit quantum gate |
| *ctrl_dits* | Classical control dits indexes |
| *target* | Target qudit indexes where the gate *U* is applied depending on the values of the classical control dits |
| *name* | Optional gate name |

**Returns**

Reference to the current instance

#### 7.54.5.10 cend()

```
const_iterator qpp::QCircuit::cend ( ) const [inline], [noexcept]
```

Constant iterator to the next to the last element.

**Returns**

Constant iterator to the next to the last element

#### 7.54.5.11 CTRL() [1/4]

```
QCircuit& qpp::QCircuit::CTRL (
            const cmat & U,
            idx ctrl,
            idx target,
            std::string name = {} )  [inline]
```

Applies the single qudit controlled gate *U* with control qudit *ctrl* and target qudit *target*.

**Parameters**

| | |
|---|---|
| *U* | Single qudit quantum gate |
| *ctrl* | Control qudit index |
| *target* | Target qudit index |
| *name* | Optional gate name |

**Returns**

Reference to the current instance

#### 7.54.5.12 CTRL() [2/4]

```
QCircuit& qpp::QCircuit::CTRL (
            const cmat & U,
            idx ctrl,
            const std::vector< idx > & target,
            std::string name = {} )  [inline]
```

Applies the single qudit controlled gate *U* with control qudit *ctrl* on every qudit listed in *target*.

**Parameters**

| | |
|---|---|
| *U* | Single qudit quantum gate |
| *ctrl* | Control qudit index |
| *target* | Target qudit indexes; the gate *U* is applied on every one of them depending on the values of the control qudits |
| *name* | Optional gate name |

**Returns**

Reference to the current instance

#### 7.54.5.13 CTRL() [3/4]

```
QCircuit& qpp::QCircuit::CTRL (
            const cmat & U,
            const std::vector< idx > & ctrl,
            idx target,
            std::string name = {} ) [inline]
```

Applies the single qudit controlled gate *U* with multiple control qudits listed in *ctrl* on the target qudit *target*.

**Parameters**

| | |
|---|---|
| *U* | Single qudit quantum gate |
| *ctrl* | Control qudit indexes |
| *target* | Target qudit index |
| *name* | Optional gate name |

**Returns**

Reference to the current instance

#### 7.54.5.14 CTRL() [4/4]

```
QCircuit& qpp::QCircuit::CTRL (
            const cmat & U,
            const std::vector< idx > & ctrl,
            const std::vector< idx > & target,
            std::string name = {} ) [inline]
```

Applies the single qudit controlled gate *U* with multiple control qudits listed in *ctrl* on every qudit listed in *target*.

**Parameters**

| | |
|---|---|
| *U* | Single qudit quantum gate |
| *ctrl* | Control qudit indexes |
| *target* | Target qudit indexes; the gate *U* is applied on every one of them depending on the values of the control qudits |
| *name* | Optional gate name |

**Returns**

Reference to the current instance

#### 7.54.5.15 CTRL_custom()

```
QCircuit& qpp::QCircuit::CTRL_custom (
            const cmat & U,
            const std::vector< idx > & ctrl,
            const std::vector< idx > & target,
            std::string name = {} )  [inline]
```

Jointly applies the custom multiple-qudit controlled gate *U* with multiple control qudits listed in *ctrl* on the qudit indexes specified by *target*.

**Parameters**

| | |
|---|---|
| *U* | Multiple-qudit quantum gate |
| *ctrl* | Control qudit indexes |
| *target* | Target qudit indexes where the gate *U* is applied depending on the values of the control qudits |
| *name* | Optional gate name |

**Returns**

Reference to the current instance

#### 7.54.5.16 display()

```
std::ostream& qpp::QCircuit::display (
            std::ostream & os ) const  [inline], [override], [private], [virtual]
```

qpp::IDisplay::display() override

Writes to the output stream a textual representation of the quantum circuit

**Parameters**

| | |
|---|---|
| *os* | Output stream passed by reference |

**Returns**

Reference to the output stream

Implements qpp::IDisplay.

#### 7.54.5.17 end() [1/2]

```
iterator qpp::QCircuit::end ( ) [inline]
```

Iterator to the next to the last element.

**Returns**

Iterator to the next to the last element

#### 7.54.5.18 end() [2/2]

```
const_iterator qpp::QCircuit::end ( ) const [inline], [noexcept]
```

Constant iterator to the next to the last element.

**Returns**

Constant iterator to the next to the last element

#### 7.54.5.19 gate() [1/3]

```
QCircuit& qpp::QCircuit::gate (
            const cmat & U,
            idx i,
            std::string name = {} ) [inline]
```

Applies the single qudit gate *U* on single qudit *i*.

**Parameters**

| | |
|---|---|
| *U* | Single qudit quantum gate |
| *i* | Qudit index |
| *name* | Optional gate name |

**Returns**

Reference to the current instance

#### 7.54.5.20 gate() [2/3]

```
QCircuit& qpp::QCircuit::gate (
            const cmat & U,
```

```
            idx i,
            idx j,
            std::string name = {} )  [inline]
```

Applies the two qudit gate *U* on qudits *i* and *j*.

**Parameters**

| | |
|---|---|
| *U* | Two qudit quantum gate |
| *i* | Qudit index |
| *j* | Qudit index |
| *name* | Optional gate name |

**Returns**

Reference to the current instance

#### 7.54.5.21 gate() [3/3]

```
QCircuit& qpp::QCircuit::gate (
            const cmat & U,
            idx i,
            idx j,
            idx k,
            std::string name = {} )  [inline]
```

Applies the three qudit gate *U* on qudits *i*, *j* and *k*.

**Parameters**

| | |
|---|---|
| *U* | Three qudit quantum gate |
| *i* | Qudit index |
| *j* | Qudit index |
| *k* | Qudit index |
| *name* | Optional gate name |

**Returns**

Reference to the current instance

#### 7.54.5.22 gate_custom()

```
QCircuit& qpp::QCircuit::gate_custom (
            const cmat & U,
            const std::vector< idx > & target,
            std::string name = {} )  [inline]
```

Jointly applies the custom multiple qudit gate *U* on the qudit indexes specified by *target*.

**Parameters**

| | |
|---|---|
| *U* | Multiple qudit quantum gate |
| *target* | Subsystem indexes where the gate *U* is applied |
| *name* | Optional gate name |

**Returns**

Reference to the current instance

#### 7.54.5.23 gate_fan() [1/3]

```
QCircuit& qpp::QCircuit::gate_fan (
        const cmat & U,
        const std::vector< idx > & target,
        std::string name = {} ) [inline]
```

Applies the single qudit gate *U* on every qudit listed in *target*.

**Parameters**

| | |
|---|---|
| *U* | Single qudit quantum gate |
| *target* | Target qudit indexes; the gate *U* is applied on every one of them |
| *name* | Optional gate name |

**Returns**

Reference to the current instance

#### 7.54.5.24 gate_fan() [2/3]

```
QCircuit& qpp::QCircuit::gate_fan (
        const cmat & U,
        const std::initializer_list< idx > & target,
        std::string name = {} ) [inline]
```

Applies the single qudit gate *U* on every qudit listed in *target*.

**Parameters**

| | |
|---|---|
| *U* | Single qudit quantum gate |
| *target* | Target qudit indexes; the gate *U* is applied on every one of them |
| *name* | Optional gate name |

**Returns**

Reference to the current instance

#### 7.54.5.25 gate_fan() [3/3]

```
QCircuit& qpp::QCircuit::gate_fan (
            const cmat & U,
            std::string name = {} )  [inline]
```

Applies the single qudit gate *U* on every remaining non-measured qudit.

**Parameters**

| | |
|---|---|
| *U* | Single qudit quantum gate |
| *name* | Optional gate name |

**Returns**

Reference to the current instance

#### 7.54.5.26 get_cmat_hash_tbl_()

```
const std::unordered_map<std::size_t, cmat>& qpp::QCircuit::get_cmat_hash_tbl_ ( ) const
[inline], [private], [noexcept]
```

Hash table with the matrices used in the circuit.

**Returns**

Hash table with the matrices used in the circuit

#### 7.54.5.27 get_d()

```
idx qpp::QCircuit::get_d ( ) const  [inline], [noexcept]
```

Dimension of the comprising qudits.

**Returns**

Qudit dimension

#### 7.54.5.28 get_gate_count()

```
idx qpp::QCircuit::get_gate_count (
            const std::string & name = {} ) const  [inline]
```

Quantum circuit gate count.

**Note**

If *name* is empty (default), returns the total gate count of the circuit

**Parameters**

| | |
|---|---|
| *name* | Gate name (optional) |

**Returns**

Gate count

#### 7.54.5.29 get_gate_depth()

```
idx qpp::QCircuit::get_gate_depth (
            const std::string & name = {} ) const  [inline]
```

Quantum circuit gate depth.

**Note**

If *name* is empty (default), returns the total gate depth of the circuit

**Parameters**

| | |
|---|---|
| *name* | Gate name (optional) |

**Returns**

Gate depth

#### 7.54.5.30 get_gates_()

```
const std::vector<GateStep>& qpp::QCircuit::get_gates_ ( ) const  [inline], [private], [noexcept]
```

Vector of qpp::QCircuit::GateStep.

**Returns**

Vector of qpp::QCircuit::GateStep

#### 7.54.5.31 get_measured() [1/2]

```
idx qpp::QCircuit::get_measured (
            idx i ) const  [inline]
```

Check whether qudit *i* was already measured.

**Parameters**

| | |
|---|---|
| *i* | Qudit index |

**Returns**

True if qudit *i* was already measured, false othwewise

#### 7.54.5.32 get_measured() [2/2]

```
std::vector<idx> qpp::QCircuit::get_measured ( ) const  [inline]
```

Vector of already measured qudit indexes.

**Returns**

Vector of already measured qudit indexes

#### 7.54.5.33 get_measurement_count() [1/2]

```
idx qpp::QCircuit::get_measurement_count ( ) const  [inline], [noexcept]
```

Quantum circuit total measurement count.

**Returns**

Total measurement count

#### 7.54.5.34 get_measurement_count() [2/2]

```
idx qpp::QCircuit::get_measurement_count (
            const std::string & name ) const  [inline]
```

Quantum circuit measurement count.

**Parameters**

| | |
|---|---|
| *name* | Measurement name |

**Returns**

Measurement count

#### 7.54.5.35 get_measurements_()

```
const std::vector<MeasureStep>& qpp::QCircuit::get_measurements_ ( ) const [inline], [private],
[noexcept]
```

Vector of qpp::QCircuit::MeasureStep.

**Returns**

Vector of qpp::QCircuit::MeasureStep

#### 7.54.5.36 get_name()

```
std::string qpp::QCircuit::get_name ( ) const [inline]
```

Quantum circuit name.

**Returns**

Quantum circuit name

#### 7.54.5.37 get_nc()

```
idx qpp::QCircuit::get_nc ( ) const [inline], [noexcept]
```

Total number of classical dits in the circuit.

**Returns**

Total number of classical dits

#### 7.54.5.38 get_non_measured()

```
std::vector<idx> qpp::QCircuit::get_non_measured ( ) const [inline]
```

Vector of non-measured qudit indexes.

**Returns**

Vector of non-measured qudit indexes

#### 7.54.5.39 get_nop_count()

```
idx qpp::QCircuit::get_nop_count ( ) const [inline]
```

No-op count.

**Returns**

No-op count

#### 7.54.5.40 get_nq()

```
idx qpp::QCircuit::get_nq ( ) const [inline], [noexcept]
```

Total number of qudits in the circuit.

**Returns**

Total number of qudits

#### 7.54.5.41 get_step_count()

```
idx qpp::QCircuit::get_step_count ( ) const [inline], [noexcept]
```

Quantum circuit total steps count, i.e. the sum of gate count and measurement count.

**Returns**

Total (gates + measurements) count

#### 7.54.5.42 measureV() [1/2]

```
QCircuit& qpp::QCircuit::measureV (
        const cmat & V,
        idx target,
        idx c_reg,
        std::string name = {} ) [inline]
```

Measurement of single qudit in the orthonormal basis or rank-1 projectors specified by the columns of matrix *V*.

**Parameters**

| | |
|---|---|
| *V* | Orthonormal basis or rank-1 projectors specified by the columns of matrix V |
| *target* | Qudit index |
| *c_reg* | Classical register where the value of the measurement is stored |
| *name* | Optional measurement name |

**Returns**

Reference to the current instance

#### 7.54.5.43 measureV() [2/2]

```
QCircuit& qpp::QCircuit::measureV (
            const cmat & V,
            const std::vector< idx > & target,
            idx c_reg,
            std::string name = {} )  [inline]
```

Joint measurement of multiple qudits in the orthonormal basis or rank-1 projectors specified by the columns of matrix *V*.

**Parameters**

| | |
|---|---|
| *V* | Orthonormal basis or rank-1 projectors specified by the columns of matrix V |
| *target* | Target qudit indexes that are jointly measured |
| *c_reg* | Classical register where the value of the measurement is stored |
| *name* | Optional measurement name |

**Returns**

Reference to the current instance

#### 7.54.5.44 measureZ()

```
QCircuit& qpp::QCircuit::measureZ (
            idx target,
            idx c_reg,
            std::string name = {} )  [inline]
```

Measurement of single qudit in the computational basis (Z-basis)

**Parameters**

| | |
|---|---|
| *target* | Qudit index |
| *c_reg* | Classical register where the value of the measurement is being stored |
| *name* | Optional measurement name, default is "Measure Z" |

**Returns**

Reference to the current instance

#### 7.54.5.45 nop()

```
QCircuit& qpp::QCircuit::nop ( ) [inline]
```

No operation (no-op)

**Note**

If the underlying step is executed on a noisy engine, then noise acts before it

**Returns**

Reference to the current instance

#### 7.54.5.46 QFT() [1/3]

```
QCircuit& qpp::QCircuit::QFT (
            const std::vector< idx > & target,
            bool swap = true ) [inline]
```

Applies the quantum Fourier transform (as a series of gates) on the qudit indexes specified by *target*.

**Parameters**

| | |
|---|---|
| *target* | Subsystem indexes where the quantum Fourier transform is applied |
| *swap* | Swaps the qubits at the end (true by default) |

**Returns**

Reference to the current instance

#### 7.54.5.47 QFT() [2/3]

```
QCircuit& qpp::QCircuit::QFT (
            const std::initializer_list< idx > & target,
            bool swap = true ) [inline]
```

Applies the quantum Fourier transform (as a series of gates) on the qudit indexes specified by *target*.

**Parameters**

| | |
|---|---|
| *target* | Subsystem indexes where the quantum Fourier transform is applied |
| *swap* | Swaps the qubits at the end (true by default) |

**Returns**

Reference to the current instance

#### 7.54.5.48 QFT() [3/3]

```
QCircuit& qpp::QCircuit::QFT (
            bool swap = true )  [inline]
```

Applies the quantum Fourier transform (as a series of gates) on all of remaining non-measured qudits.

**Parameters**

| | |
|---|---|
| *swap* | Swaps the qubits at the end (true by default) |

**Returns**

Reference to the current instance

#### 7.54.5.49 TFQ() [1/3]

```
QCircuit& qpp::QCircuit::TFQ (
            const std::vector< idx > & target,
            bool swap QPP_UNUSED_ = true )  [inline]
```

Applies the inverse quantum Fourier transform (as a series of gates) on the qudit indexes specified by *target*.

**Parameters**

| | |
|---|---|
| *target* | Subsystem indexes where the inverse quantum Fourier transform is applied |
| *swap* | Swaps the qubits at the end (true by default) |

**Returns**

Reference to the current instance

#### 7.54.5.50 TFQ() [2/3]

```
QCircuit& qpp::QCircuit::TFQ (
            const std::initializer_list< idx > & target,
            bool swap = true )  [inline]
```

Applies the inverse quantum Fourier transform (as a series of gates) on the qudit indexes specified by *target*.

**Parameters**

| | |
|---|---|
| *target* | Subsystem indexes where the inverse quantum Fourier transform is applied |
| *swap* | Swaps the qubits at the end (true by default) |

**Returns**

Reference to the current instance

#### 7.54.5.51 TFQ() [3/3]

```
QCircuit& qpp::QCircuit::TFQ (
            bool swap = true )  [inline]
```

Applies the inverse quantum Fourier transform (as a series of gates) on all of remaining non-measured qudits.

**Parameters**

| | |
|---|---|
| *swap* | Swaps the qubits at the end (true by default) |

**Returns**

Reference to the current instance

#### 7.54.5.52 to_JSON()

```
std::string qpp::QCircuit::to_JSON (
            bool enclosed_in_curly_brackets = true ) const  [inline], [override], [virtual]
```

qpp::IJOSN::to_JSON() override

Displays the quantum circuit in JSON format

**Parameters**

| | |
|---|---|
| *enclosed_in_curly_brackets* | If true, encloses the result in curly brackets |

**Returns**

String containing the JSON representation of the quantum circuit

Implements qpp::IJSON.

### 7.54.6 Friends And Related Function Documentation

#### 7.54.6.1 operator<< [1/4]

```
std::ostream& operator<< (
            std::ostream & os,
            const GateType & gate_type )  [friend]
```

Extraction operator overload for qpp::QCircuit::GateType enum class.

**Parameters**

| | |
|---|---|
| *os* | Output stream |
| *gate_type* | qpp::QCircuit::GateType enum class |

**Returns**

Output stream

#### 7.54.6.2 operator<< [2/4]

```
std::ostream& operator<< (
            std::ostream & os,
            const GateStep & gate_step )  [friend]
```

Extraction operator overload for qpp::QCircuit::GateStep class.

**Parameters**

| | |
|---|---|
| *os* | Output stream |
| *gate_step* | qpp::QCircuit::GateStep class |

**Returns**

Output stream

#### 7.54.6.3 operator<< [3/4]

```
std::ostream& operator<< (
            std::ostream & os,
            const MeasureType & measure_type )  [friend]
```

Extraction operator overload for qpp::QCircuit::MeasureType enum class.

**Parameters**

| | |
|---|---|
| *os* | Output stream |
| *measure_type* | qpp::QCircuit::MeasureType enum class |

**Returns**

Output stream

#### 7.54.6.4 operator<< [4/4]

```
std::ostream& operator<< (
            std::ostream & os,
            const MeasureStep & measure_step )  [friend]
```

Extraction operator overload for qpp::QCircuit::MeasureStep class.

**Parameters**

| | |
|---|---|
| *os* | Output stream |
| *measure_step* | qpp::QCircuit::MeasureStep enum class |

**Returns**

Output stream

#### 7.54.6.5 QEngine

```
friend class QEngine  [friend]
```

### 7.54.7 Member Data Documentation

#### 7.54.7.1 cmat_hash_tbl_

```
std::unordered_map<std::size_t, cmat> qpp::QCircuit::cmat_hash_tbl_ {} [private]
```

hash table with the matrices used in the circuit, with [Key = idx, Value = cmat]

#### 7.54.7.2 count_

```
std::unordered_map<std::string, idx> qpp::QCircuit::count_ {} [private]
```

gate counts

#### 7.54.7.3 d_

```
const idx qpp::QCircuit::d_ [private]
```

qudit dimension

#### 7.54.7.4 gates_

```
std::vector<GateStep> qpp::QCircuit::gates_ {} [private]
```

gates

#### 7.54.7.5 measured_

```
std::vector<bool> qpp::QCircuit::measured_ [private]
```

keeps track of the measured qudits

#### 7.54.7.6 measurement_count_

```
std::unordered_map<std::string, idx> qpp::QCircuit::measurement_count_ {} [private]
```

measurement counts

#### 7.54.7.7 measurements_

```
std::vector<MeasureStep> qpp::QCircuit::measurements_ {} [private]
```

measurements

#### 7.54.7.8 name_

```
std::string qpp::QCircuit::name_ [private]
```

optional circuit name

#### 7.54.7.9 nc_

```
const idx qpp::QCircuit::nc_ [private]
```

number of classical "dits"

#### 7.54.7.10 nq_

```
const idx qpp::QCircuit::nq_ [private]
```

number of qudits

#### 7.54.7.11 step_types_

```
std::vector<StepType> qpp::QCircuit::step_types_ {} [private]
```

type of each step

The documentation for this class was generated from the following file:

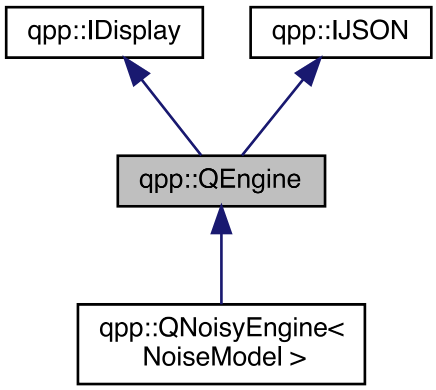
- classes/circuits/circuits.h

## 7.55 qpp::QEngine Class Reference

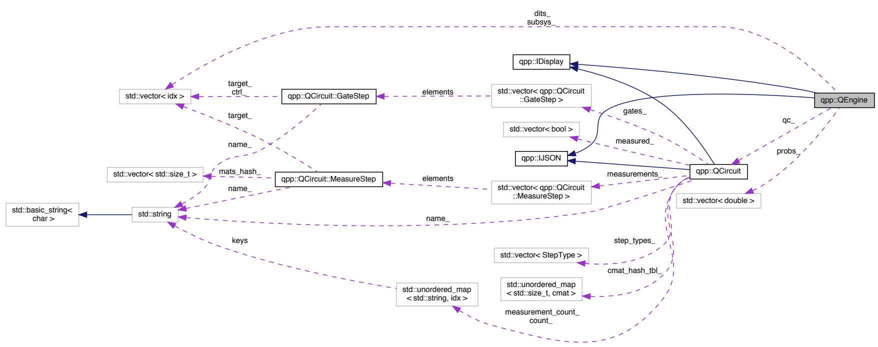
Quantum circuit engine, executes qpp::QCircuit.

```
#include <classes/circuits/engines.h>
```

Inheritance diagram for qpp::QEngine:




Collaboration diagram for qpp::QEngine:



### Public Member Functions

- QEngine (const QCircuit &qc)

  *Constructs a quantum engine out of a quantum circuit.*
- QEngine (const QEngine &)=default

  *Default copy constructor.*
- QEngine & operator= (const QEngine &)=default

  *Default copy assignment operator.*
- QEngine (QCircuit &&)=delete

*Disables rvalue QCircuit.*
- virtual ∼QEngine ()=default

  *Default virtual destructor.*
- ket get_psi () const

  *Underlying quantum state.*
- std::vector< idx > get_dits () const

  *Vector with the values of the underlying classical dits.*
- idx get_dit (idx i) const

  *Value of the classical dit at position i.*
- std::vector< double > get_probs () const

  *Vector of underlying measurement outcome probabilities.*
- bool get_measured (idx i) const

  *Check whether qudit i was already measured.*
- std::vector< idx > get_measured () const

  *Vector of already measured qudit indexes.*
- std::vector< idx > get_non_measured () const

  *Vector of non-measured qudit indexes.*
- const QCircuit & get_circuit () const noexcept

  *Quantum circuit.*
- QEngine & set_dit (idx i, idx value)

  *Sets the classical dit at position i.*
- QEngine & set_psi (const ket &psi)

  *Sets the underlying quantum state to psi.*
- void reset ()

  *Resets the engine.*
- virtual void execute (const QCircuit::iterator::value_type &elem)

  *Executes one step in the quantum circuit.*
- void execute (const QCircuit::iterator &it)

  *Executes one step in the quantum circuit.*
- void execute ()

  *Executes the entire quantum circuit.*
- std::string to_JSON (bool enclosed_in_curly_brackets=true) const override

  *qpp::IJOSN::to_JSON() override*

## Protected Member Functions

- void set_measured_ (idx i)

  *Marks qudit i as measured then re-label accordingly the remaining non-measured qudits.*
- std::vector< idx > get_relative_pos_ (std::vector< idx > v)

  *Giving a vector V of non-measured qudits, get their relative position with respect to the measured qudits.*

## Protected Attributes

- const QCircuit * qc_

  *pointer to constant quantum circuit*
- ket psi_

  *state vector*
- std::vector< idx > dits_

  *classical dits*
- std::vector< double > probs_

  *measurement probabilities*
- std::vector< idx > subsys_

  *relabel them after measurements*

### Private Member Functions

- std::ostream & display (std::ostream &os) const override
  *qpp::IDisplay::display() override*

## 7.55.1 Detailed Description

Quantum circuit engine, executes qpp::QCircuit.

**See also**

qpp::QCircuit

## 7.55.2 Constructor & Destructor Documentation

### 7.55.2.1 QEngine() [1/3]

```
qpp::QEngine::QEngine (
            const QCircuit & qc )  [inline], [explicit]
```

Constructs a quantum engine out of a quantum circuit.

**Note**

The quantum circuit must be an lvalue

**See also**

qpp::QEngine(QCircuit&&)

**Note**

The initial underlying quantum state is set to $|0\rangle^{\otimes n}$

**Parameters**

| | |
|---|---|
| *qc* | Quantum circuit |

### 7.55.2.2 QEngine() [2/3]

```
qpp::QEngine::QEngine (
            const QEngine &  )  [default]
```

Default copy constructor.

#### 7.55.2.3 QEngine() [3/3]

```
qpp::QEngine::QEngine (
            QCircuit && ) [delete]
```

Disables rvalue QCircuit.

#### 7.55.2.4 ∼QEngine()

```
virtual qpp::QEngine::∼QEngine ( ) [virtual], [default]
```

Default virtual destructor.

### 7.55.3 Member Function Documentation

#### 7.55.3.1 display()

```
std::ostream& qpp::QEngine::display (
            std::ostream & os ) const [inline], [override], [private], [virtual]
```

qpp::IDisplay::display() override

Writes to the output stream a textual representation of the state of the engine

**Parameters**

| | |
|---|---|
| *os* | Output stream passed by reference |

**Returns**

Reference to the output stream

Implements qpp::IDisplay.

#### 7.55.3.2 execute() [1/3]

```
virtual void qpp::QEngine::execute (
            const QCircuit::iterator::value_type & elem ) [inline], [virtual]
```

Executes one step in the quantum circuit.

**Parameters**

| | |
|---|---|
| *elem* | Step to be executed |

Reimplemented in qpp::QNoisyEngine< NoiseModel >.

#### 7.55.3.3 execute() [2/3]

```
void qpp::QEngine::execute (
            const QCircuit::iterator & it ) [inline]
```

Executes one step in the quantum circuit.

**Parameters**

| | |
|---|---|
| *it* | Iterator to the step to be executed |

#### 7.55.3.4 execute() [3/3]

```
void qpp::QEngine::execute ( ) [inline]
```

Executes the entire quantum circuit.

#### 7.55.3.5 get_circuit()

```
const QCircuit& qpp::QEngine::get_circuit ( ) const [inline], [noexcept]
```

Quantum circuit.

**Returns**

Underlying quantum circuit

#### 7.55.3.6 get_dit()

```
idx qpp::QEngine::get_dit (
            idx i ) const [inline]
```

Value of the classical dit at position *i*.

**Parameters**

| | |
|---|---|
| *i* | Classical dit index |

**Returns**

Value of the classical dit at position *i*

#### 7.55.3.7 get_dits()

```
std::vector<idx> qpp::QEngine::get_dits ( ) const [inline]
```

Vector with the values of the underlying classical dits.

**Returns**

Vector of underlying classical dits

#### 7.55.3.8 get_measured() [1/2]

```
bool qpp::QEngine::get_measured (
            idx i ) const [inline]
```

Check whether qudit *i* was already measured.

**Parameters**

| | |
|---|---|
| *i* | Qudit index |

**Returns**

True if qudit *i* was already measured, false othwewise

#### 7.55.3.9 get_measured() [2/2]

```
std::vector<idx> qpp::QEngine::get_measured ( ) const [inline]
```

Vector of already measured qudit indexes.

**Returns**

Vector of already measured qudit indexes

#### 7.55.3.10 get_non_measured()

```
std::vector<idx> qpp::QEngine::get_non_measured ( ) const [inline]
```

Vector of non-measured qudit indexes.

**Returns**

Vector of non-measured qudit indexes

#### 7.55.3.11 get_probs()

```
std::vector<double> qpp::QEngine::get_probs ( ) const [inline]
```

Vector of underlying measurement outcome probabilities.

Those should be interpreted as conditional probabilities based on the temporal order of the measurements, i.e. if we measure qubit 0, then measure qubit 1, and finally qubit 2, the resulting vector of outcome probabilities probs[2] should be interpreted as the conditional probability of qubit 2 having the outcome it had given that qubit 1 and qubit 0 had their given outcomes, respectively. As an example, if we measure the qubit 0 followed by the qubit 1 of a maximally entangled state $(|00\rangle + |11\rangle)/\sqrt{2}$, then the vector of outcome probabilities will be [0.5, 1].

**Note**

The probability vector has the same length as the vector of classical dits. If the measurement result is stored at the index *c_reg*, then the outcome probability is automatically stored at the same index *c_reg* in the probability vector.

**Returns**

Vector of underlying measurement outcome probabilities

#### 7.55.3.12 get_psi()

```
ket qpp::QEngine::get_psi ( ) const [inline]
```

Underlying quantum state.

**Returns**

Underlying quantum state

#### 7.55.3.13 get_relative_pos_()

```
std::vector<idx> qpp::QEngine::get_relative_pos_ (
            std::vector< idx > v ) [inline], [protected]
```

Giving a vector *V* of non-measured qudits, get their relative position with respect to the measured qudits.

**Parameters**

| | |
|---|---|
| *v* | |

**Returns**

Vector of qudit indexes

#### 7.55.3.14 operator=()

```
QEngine& qpp::QEngine::operator= (
            const QEngine & )  [default]
```

Default copy assignment operator.

**Returns**

Reference to the current instance

#### 7.55.3.15 reset()

```
void qpp::QEngine::reset ( )  [inline]
```

Resets the engine.

Re-initializes everything to zero and sets the initial state to $|0\rangle^{\otimes n}$

#### 7.55.3.16 set_dit()

```
QEngine& qpp::QEngine::set_dit (
            idx i,
            idx value )  [inline]
```

Sets the classical dit at position *i*.

**Parameters**

| | |
|---|---|
| *i* | Classical dit index |
| *value* | Classical dit value |

**Returns**

Reference to the current instance

#### 7.55.3.17 set_measured_()

```
void qpp::QEngine::set_measured_ (
            idx i ) [inline], [protected]
```

Marks qudit *i* as measured then re-label accordingly the remaining non-measured qudits.

**Parameters**

| | |
|---|---|
| *i* | Qudit index |

#### 7.55.3.18 set_psi()

```
QEngine& qpp::QEngine::set_psi (
            const ket & psi ) [inline]
```

Sets the underlying quantum state to *psi*.

**Note**

The order is lexicographical with respect to the remaining non-measured qudits

**Parameters**

| | |
|---|---|
| *psi* | State vector |

**Returns**

Reference to the current instance

#### 7.55.3.19 to_JSON()

```
std::string qpp::QEngine::to_JSON (
            bool enclosed_in_curly_brackets = true ) const [inline], [override], [virtual]
```

qpp::IJOSN::to_JSON() override

Displays the state of the engine in JSON format

**Parameters**

| | |
|---|---|
| *enclosed_in_curly_brackets* | If true, encloses the result in curly brackets |

**Returns**

String containing the JSON representation of the state of the engine

Implements qpp::IJSON.

### 7.55.4 Member Data Documentation

#### 7.55.4.1 dits_

```
std::vector<idx> qpp::QEngine::dits_  [protected]
```

classical dits

#### 7.55.4.2 probs_

```
std::vector<double> qpp::QEngine::probs_  [protected]
```

measurement probabilities

#### 7.55.4.3 psi_

```
ket qpp::QEngine::psi_  [protected]
```

state vector

#### 7.55.4.4 qc_

```
const QCircuit* qpp::QEngine::qc_  [protected]
```

pointer to constant quantum circuit

#### 7.55.4.5 subsys_

```
std::vector<idx> qpp::QEngine::subsys_ [protected]
```

relabel them after measurements

keeps track of the measured subsystems,

The documentation for this class was generated from the following file:

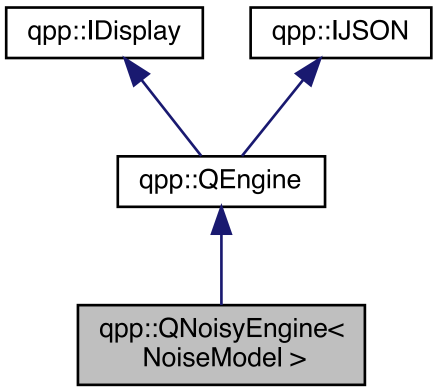
- classes/circuits/engines.h

## 7.56 qpp::QNoisyEngine< NoiseModel > Class Template Reference

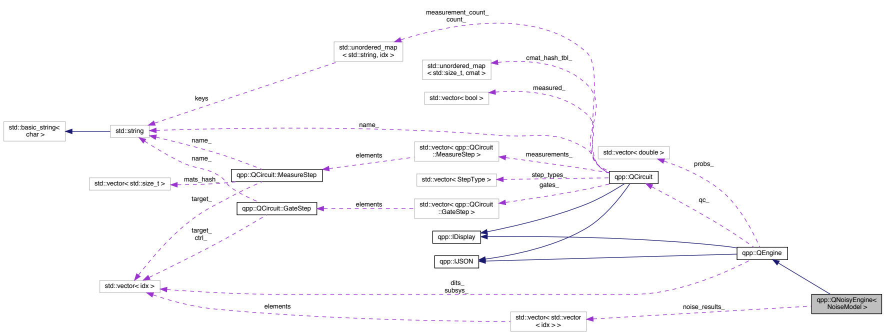
Noisy quantum circuit engine, executes qpp::QCircuit.

```
#include <classes/circuits/engines.h>
```

Inheritance diagram for qpp::QNoisyEngine< NoiseModel >:




Collaboration diagram for qpp::QNoisyEngine< NoiseModel >:



## Public Member Functions

- QNoisyEngine (const QCircuit &qc, const NoiseModel &noise)
  *Constructs a noisy quantum engine out of a quantum circuit.*
- void execute (const QCircuit::iterator::value_type &elem) override
  *Executes one step in the quantum circuit.*
- std::vector< std::vector< idx > > get_noise_results () const
  *Vector of noise results obtained before every step in the circuit.*
- virtual void execute (const QCircuit::iterator::value_type &elem)
  *Executes one step in the quantum circuit.*
- void execute (const QCircuit::iterator &it)
  *Executes one step in the quantum circuit.*
- void execute ()
  *Executes the entire quantum circuit.*

## Private Attributes

- const NoiseModel noise_
  *quantum noise model*
- std::vector< std::vector< idx > > noise_results_
  *noise results*

## Additional Inherited Members

### 7.56.1 Detailed Description

**template<typename NoiseModel>**
**class qpp::QNoisyEngine< NoiseModel >**

Noisy quantum circuit engine, executes qpp::QCircuit.

**See also**

qpp::QCircuit, qpp::NoiseBase

Assumes an uncorrelated noise model that is applied to each non-measured qubit before every step in the logical circuit

**Template Parameters**

| | |
|---|---|
| *NoiseModel* | Quantum noise model, should be derived from qpp::NoiseBase |

### 7.56.2 Constructor & Destructor Documentation

#### 7.56.2.1 QNoisyEngine()

```
template<typename NoiseModel >
qpp::QNoisyEngine< NoiseModel >::QNoisyEngine (
            const QCircuit & qc,
            const NoiseModel & noise ) [inline], [explicit]
```

Constructs a noisy quantum engine out of a quantum circuit.

**Parameters**

| | |
|---|---|
| *qc* | Quantum circuit |
| *noise* | Quantum noise model |

### 7.56.3 Member Function Documentation

#### 7.56.3.1 execute() [1/4]

```
template<typename NoiseModel >
virtual void qpp::QEngine::execute [inline]
```

Executes one step in the quantum circuit.

**Parameters**

| | |
|---|---|
| *elem* | Step to be executed |

#### 7.56.3.2 execute() [2/4]

```
template<typename NoiseModel >
void qpp::QEngine::execute [inline]
```

Executes the entire quantum circuit.

#### 7.56.3.3 execute() [3/4]

```
template<typename NoiseModel >
void qpp::QEngine::execute [inline]
```

Executes one step in the quantum circuit.

**Parameters**

| | |
|---|---|
| *it* | Iterator to the step to be executed |

#### 7.56.3.4 execute() [4/4]

```
template<typename NoiseModel >
void qpp::QNoisyEngine< NoiseModel >::execute (
            const QCircuit::iterator::value_type & elem )  [inline], [override], [virtual]
```

Executes one step in the quantum circuit.

**Parameters**

| | |
|---|---|
| *elem* | Step to be executed |

Reimplemented from qpp::QEngine.

#### 7.56.3.5 get_noise_results()

```
template<typename NoiseModel >
std::vector<std::vector<idx> > qpp::QNoisyEngine< NoiseModel >::get_noise_results ( ) const
[inline]
```

Vector of noise results obtained before every step in the circuit.

The first vector contains the noise measurement results obtained before applying the first step in the circuit, and so on, ordered by non-measured qudits. That is, the first element in the vector corresponding to noise obtained before a given step in the circuit represents the noise result obtained on the first non-measured qudit etc.

**Returns**

Vector of noise results

### 7.56.4 Member Data Documentation

#### 7.56.4.1 noise_

```
template<typename NoiseModel >
const NoiseModel qpp::QNoisyEngine< NoiseModel >::noise_  [private]
```

quantum noise model

#### 7.56.4.2 noise_results_

```
template<typename NoiseModel >
std::vector<std::vector<idx> > qpp::QNoisyEngine< NoiseModel >::noise_results_ [private]
```

noise results

The documentation for this class was generated from the following file:

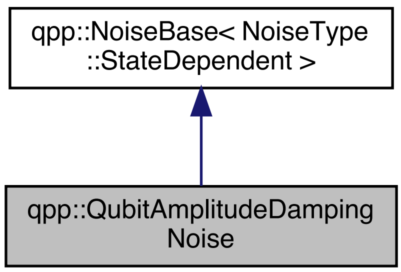
- classes/circuits/engines.h

## 7.57 qpp::QubitAmplitudeDampingNoise Class Reference

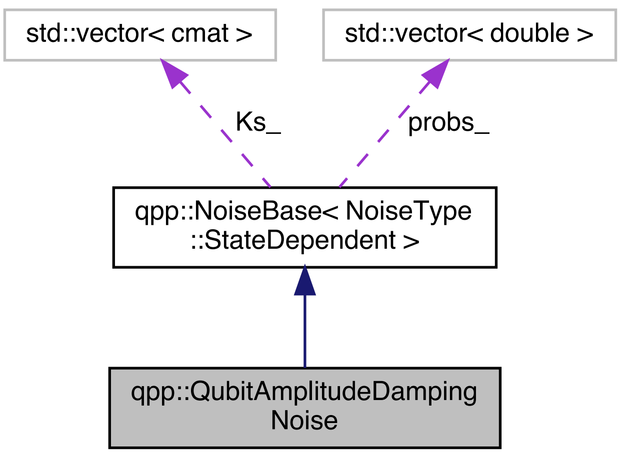
Qubit amplitude damping noise, as described in Nielsen and Chuang.

```
#include <classes/noise.h>
```

Inheritance diagram for qpp::QubitAmplitudeDampingNoise:




Collaboration diagram for qpp::QubitAmplitudeDampingNoise:



## Public Member Functions

- QubitAmplitudeDampingNoise (double gamma)

    *Qubit amplitude damping noise constructor.*

## Additional Inherited Members

### 7.57.1 Detailed Description

Qubit amplitude damping noise, as described in Nielsen and Chuang.

### 7.57.2 Constructor & Destructor Documentation

#### 7.57.2.1 QubitAmplitudeDampingNoise()

```
qpp::QubitAmplitudeDampingNoise::QubitAmplitudeDampingNoise (
            double gamma )  [inline], [explicit]
```

Qubit amplitude damping noise constructor.

**Parameters**

| | |
|---|---|
| *gamma* | Amplitude damping coefficient |

The documentation for this class was generated from the following file:

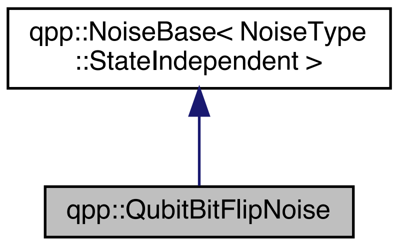
- classes/noise.h

## 7.58 qpp::QubitBitFlipNoise Class Reference

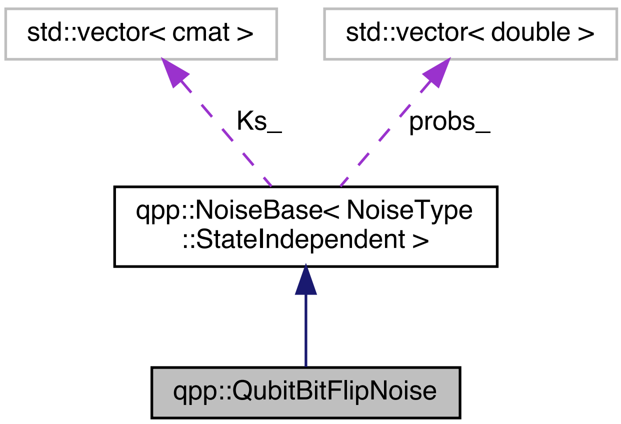
Qubit bit flip noise.

```
#include <classes/noise.h>
```

Inheritance diagram for qpp::QubitBitFlipNoise:




Collaboration diagram for qpp::QubitBitFlipNoise:




### Public Member Functions

- QubitBitFlipNoise (double p)

    *Qubit bit flip noise constructor.*

### Additional Inherited Members

### 7.58.1 Detailed Description

Qubit bit flip noise.

### 7.58.2 Constructor & Destructor Documentation

#### 7.58.2.1 QubitBitFlipNoise()

```
qpp::QubitBitFlipNoise::QubitBitFlipNoise (
            double p )  [inline], [explicit]
```

Qubit bit flip noise constructor.

**Parameters**

| | |
|---|---|
| *p* | Noise probability |

The documentation for this class was generated from the following file:

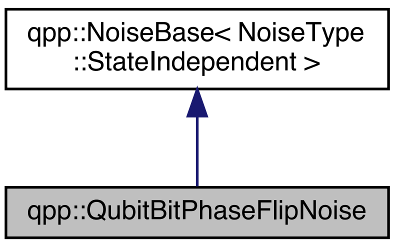
- classes/noise.h

## 7.59 qpp::QubitBitPhaseFlipNoise Class Reference

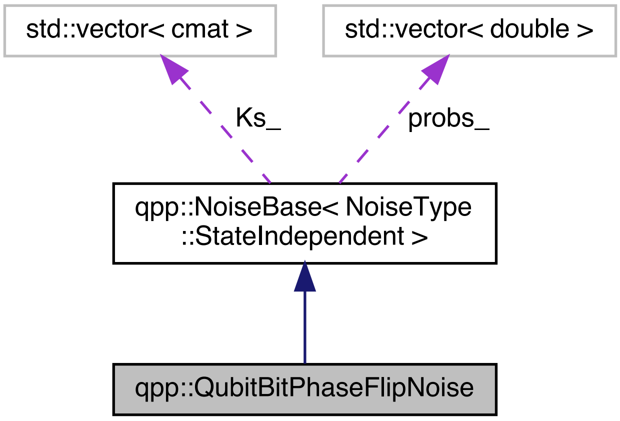
Qubit bit-phase flip (dephasing) noise.

```
#include <classes/noise.h>
```

Inheritance diagram for qpp::QubitBitPhaseFlipNoise:



Collaboration diagram for qpp::QubitBitPhaseFlipNoise:




## Public Member Functions

- QubitBitPhaseFlipNoise (double p)

  *Qubit bit-phase flip noise constructor.*

## Additional Inherited Members

### 7.59.1 Detailed Description

Qubit bit-phase flip (dephasing) noise.

### 7.59.2 Constructor & Destructor Documentation

#### 7.59.2.1 QubitBitPhaseFlipNoise()

```
qpp::QubitBitPhaseFlipNoise::QubitBitPhaseFlipNoise (
             double p )  [inline], [explicit]
```

Qubit bit-phase flip noise constructor.

**Parameters**

| | |
|---|---|
| *p* | Noise probability |

The documentation for this class was generated from the following file:

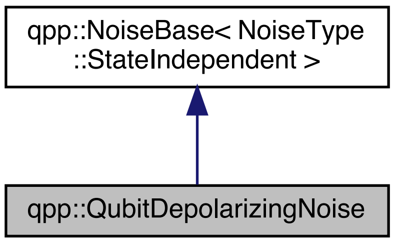
- classes/noise.h

## 7.60 qpp::QubitDepolarizingNoise Class Reference

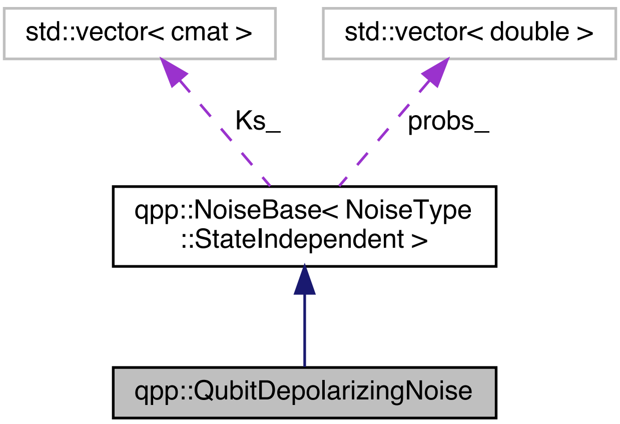
Qubit depolarizing noise.

```
#include <classes/noise.h>
```

Inheritance diagram for qpp::QubitDepolarizingNoise:




Collaboration diagram for qpp::QubitDepolarizingNoise:




### Public Member Functions

- QubitDepolarizingNoise (double p)
  *Qubit depolarizing noise constructor.*

### Additional Inherited Members

### 7.60.1 Detailed Description

Qubit depolarizing noise.

### 7.60.2 Constructor & Destructor Documentation

#### 7.60.2.1 QubitDepolarizingNoise()

```
qpp::QubitDepolarizingNoise::QubitDepolarizingNoise (
            double p )  [inline], [explicit]
```

Qubit depolarizing noise constructor.

**Parameters**

| | |
|---|---|
| *p* | Noise probability |

The documentation for this class was generated from the following file:

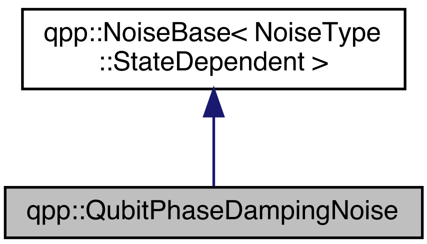
- classes/noise.h

## 7.61 qpp::QubitPhaseDampingNoise Class Reference

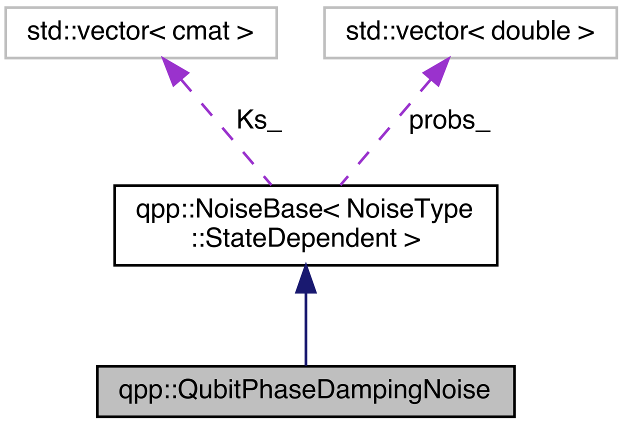
Qubit phase damping noise, as described in Nielsen and Chuang.

```
#include <classes/noise.h>
```

Inheritance diagram for qpp::QubitPhaseDampingNoise:



Collaboration diagram for qpp::QubitPhaseDampingNoise:




## Public Member Functions

- QubitPhaseDampingNoise (double lambda)
  *Qubit phase damping noise constructor.*

## Additional Inherited Members

### 7.61.1 Detailed Description

Qubit phase damping noise, as described in Nielsen and Chuang.

### 7.61.2 Constructor & Destructor Documentation

#### 7.61.2.1 QubitPhaseDampingNoise()

```
qpp::QubitPhaseDampingNoise::QubitPhaseDampingNoise (
            double lambda ) [inline], [explicit]
```

Qubit phase damping noise constructor.

**Parameters**

| | |
|---|---|
| *lambda* | Phase damping coefficient |

The documentation for this class was generated from the following file:

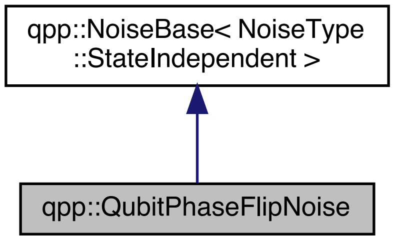
- classes/noise.h

## 7.62 qpp::QubitPhaseFlipNoise Class Reference

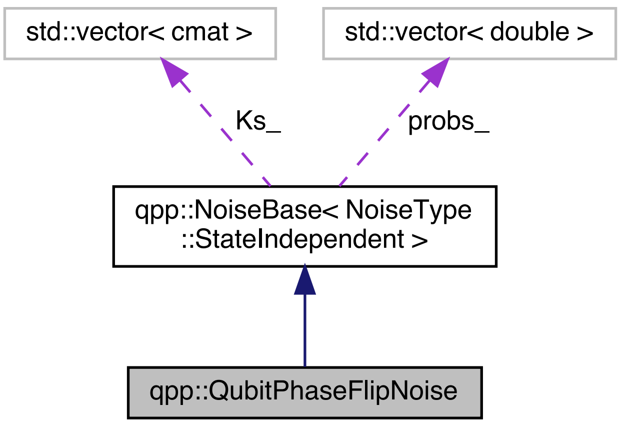
Qubit phase flip (dephasing) noise.

`#include <classes/noise.h>`

Inheritance diagram for qpp::QubitPhaseFlipNoise:




Collaboration diagram for qpp::QubitPhaseFlipNoise:




### Public Member Functions

- QubitPhaseFlipNoise (double p)

  *Qubit phase flip (dephasing) noise constructor.*

## Additional Inherited Members

### 7.62.1 Detailed Description

Qubit phase flip (dephasing) noise.

### 7.62.2 Constructor & Destructor Documentation

#### 7.62.2.1 QubitPhaseFlipNoise()

```
qpp::QubitPhaseFlipNoise::QubitPhaseFlipNoise (
             double p )  [inline], [explicit]
```

Qubit phase flip (dephasing) noise constructor.

**Parameters**

| | |
|---|---|
| *p* | Noise probability |

The documentation for this class was generated from the following file:

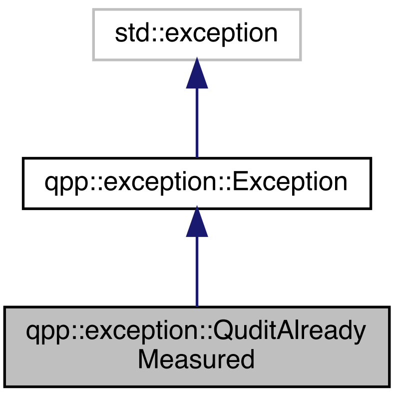
- classes/noise.h

## 7.63 qpp::exception::QuditAlreadyMeasured Class Reference

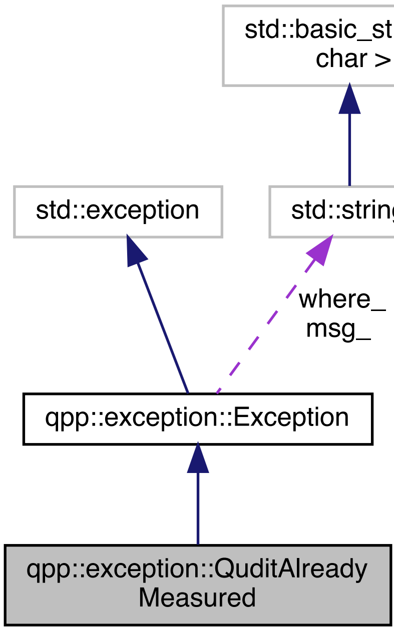
Qudit was already measured exception.

`#include <classes/exception.h>`

Inheritance diagram for qpp::exception::QuditAlreadyMeasured:



Collaboration diagram for qpp::exception::QuditAlreadyMeasured:




### Public Member Functions

- std::string type_description () const override
  *Exception type description.*
- Exception (const std::string &where)
  *Constructs an exception.*

## 7.63.1 Detailed Description

Qudit was already measured exception.

The qudit was already measured and cannot be measured again

## 7.63.2 Member Function Documentation

### 7.63.2.1 Exception()

`qpp::exception::Exception::Exception [inline], [explicit]`

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.63.2.2 type_description()

`std::string qpp::exception::QuditAlreadyMeasured::type_description ( ) const [inline], [override], [virtual]`

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

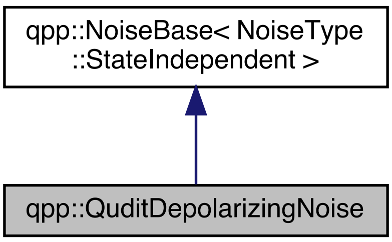
- classes/exception.h

## 7.64 qpp::QuditDepolarizingNoise Class Reference

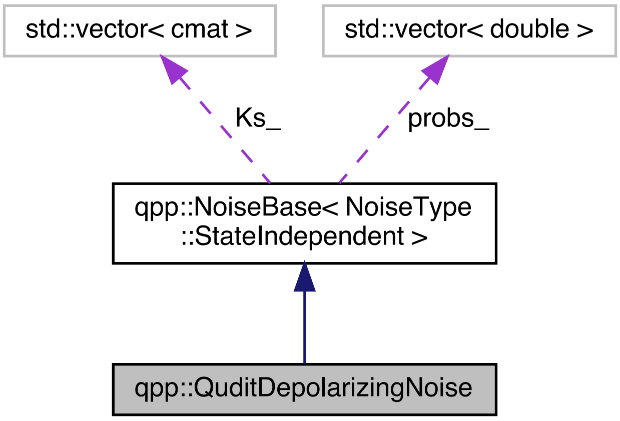
Qudit depolarizing noise.

`#include <classes/noise.h>`

Inheritance diagram for qpp::QuditDepolarizingNoise:



Collaboration diagram for qpp::QuditDepolarizingNoise:




### Public Member Functions

- QuditDepolarizingNoise (double p, idx d)

  *Qudit depolarizing noise constructor.*

### Private Member Functions

- std::vector< cmat > fill_Ks_ (idx d) const

  *Fills the Kraus operator vector.*
- std::vector< double > fill_probs_ (double p, idx d) const

  *Fills the probability vector.*

### Additional Inherited Members

## 7.64.1 Detailed Description

Qudit depolarizing noise.

## 7.64.2 Constructor & Destructor Documentation

### 7.64.2.1 QuditDepolarizingNoise()

```
qpp::QuditDepolarizingNoise::QuditDepolarizingNoise (
            double p,
            idx d ) [inline], [explicit]
```

Qudit depolarizing noise constructor.

**Parameters**

| | |
|---|---|
| *p* | Noise probability |
| *d* | Qudit dimension |

### 7.64.3 Member Function Documentation

#### 7.64.3.1 fill_Ks_()

```
std::vector<cmat> qpp::QuditDepolarizingNoise::fill_Ks_ (
            idx d ) const [inline], [private]
```

Fills the Kraus operator vector.

**Parameters**

| | |
|---|---|
| *d* | Qudit dimension |

**Returns**

Vector of Kraus operators representing the depolarizing noise

#### 7.64.3.2 fill_probs_()

```
std::vector<double> qpp::QuditDepolarizingNoise::fill_probs_ (
            double p,
            idx d ) const [inline], [private]
```

Fills the probability vector.

**Parameters**

| | |
|---|---|
| *p* | Probability |
| *d* | Qudit dimension |

**Returns**

Probability vector

The documentation for this class was generated from the following file:

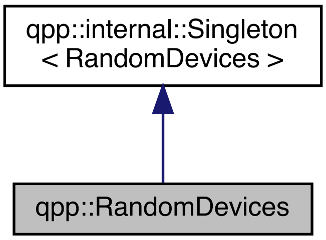
- classes/noise.h

## 7.65 qpp::RandomDevices Class Reference

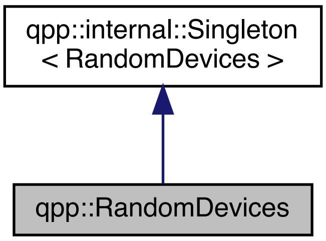
Singleton class that manages the source of randomness in the library.

`#include <classes/random_devices.h>`

Inheritance diagram for qpp::RandomDevices:




Collaboration diagram for qpp::RandomDevices:




### Public Member Functions

- std::mt19937 & get_prng ()

  *Returns a reference to the internal PRNG object.*
- std::istream & load (std::istream &is)

  *Loads the state of the PRNG from an input stream.*
- std::ostream & save (std::ostream &os) const

  *Saves the state of the PRNG to an output stream.*

## Private Member Functions

- RandomDevices ()

  *Initializes and seeds the random number generators.*
- ∼RandomDevices ()=default

  *Default destructor.*

## Private Attributes

- std::random_device rd_

  *used to seed std::mt19937 prng_*
- std::mt19937 prng_

  *Mersenne twister random number generator.*

## Friends

- class internal::Singleton< RandomDevices >

## Additional Inherited Members

### 7.65.1 Detailed Description

Singleton class that manages the source of randomness in the library.

Consists of a wrapper around an std::mt19937 Mersenne twister random number generator engine and an std↩::random_device engine. The latter is used to seed the Mersenne twister.

**Warning**

This class DOES NOT seed the standard C number generator used by Eigen::Matrix::Random(), since it is not thread safe. Do not use Eigen::Matrix::Random() or functions that depend on the C style random number engine, but use qpp::rand() instead!

### 7.65.2 Constructor & Destructor Documentation

#### 7.65.2.1 RandomDevices()

```
qpp::RandomDevices::RandomDevices ( )  [inline], [private]
```

Initializes and seeds the random number generators.

#### 7.65.2.2 ∼RandomDevices()

```
qpp::RandomDevices::∼RandomDevices ( ) [private], [default]
```

Default destructor.

### 7.65.3 Member Function Documentation

#### 7.65.3.1 get_prng()

```
std::mt19937& qpp::RandomDevices::get_prng ( ) [inline]
```

Returns a reference to the internal PRNG object.

**Returns**

Reference to the internal PRNG object

#### 7.65.3.2 load()

```
std::istream& qpp::RandomDevices::load (
            std::istream & is ) [inline]
```

Loads the state of the PRNG from an input stream.

**Parameters**

| | |
|---|---|
| *is* | Input stream |

**Returns**

The input stream

#### 7.65.3.3 save()

```
std::ostream& qpp::RandomDevices::save (
            std::ostream & os ) const [inline]
```

Saves the state of the PRNG to an output stream.

**Parameters**

| | |
|---|---|
| *os* | Output stream |

**Returns**

The output stream

### 7.65.4 Friends And Related Function Documentation

#### 7.65.4.1 internal::Singleton< RandomDevices >

```
friend class internal::Singleton< RandomDevices > [friend]
```

### 7.65.5 Member Data Documentation

#### 7.65.5.1 prng_

```
std::mt19937 qpp::RandomDevices::prng_ [private]
```

Mersenne twister random number generator.

#### 7.65.5.2 rd_

```
std::random_device qpp::RandomDevices::rd_ [private]
```

used to seed std::mt19937 prng_

The documentation for this class was generated from the following file:

- classes/random_devices.h

## 7.66 qpp::internal::Singleton< T > Class Template Reference

Singleton policy class, used internally to implement the singleton pattern via CRTP (Curiously recurring template pattern)

```
#include <internal/classes/singleton.h>
```

### Static Public Member Functions

- static T & get_instance () noexcept(std::is_nothrow_constructible< T >::value)
- static T & get_thread_local_instance () noexcept(std::is_nothrow_constructible< T >::value)

### Protected Member Functions

- Singleton () noexcept=default
- Singleton (const Singleton &)=delete
- Singleton & operator= (const Singleton &)=delete
- virtual ~Singleton ()=default

## 7.66.1 Detailed Description

**template<typename T>**
**class qpp::internal::Singleton< T >**

Singleton policy class, used internally to implement the singleton pattern via CRTP (Curiously recurring template pattern)

To implement a singleton, derive your class from qpp::internal::Singleton, make qpp::internal::Singleton a friend of your class, then declare the constructor and destructor of your class as private. To get an instance, use the static member function qpp::internal::Singleton::get_instance() (qpp::internal::Singleton::get_thread_local_instance()), which returns a reference (thread_local reference) to your newly created singleton (thread-safe in C++11).

Example:

```
class MySingleton: public qpp::internal::Singleton<MySingleton>
{
       friend class qpp::internal::Singleton<MySingleton>;
public:
    // Declare all public members here
private:
    MySingleton()
    {
        // Implement the constructor here
    }
    ~MySingleton()
    {
        // Implement the destructor here
    }
};

MySingleton& mySingleton = MySingleton::get_instance(); // Get an instance
thread_local MySingleton& tls = MySingleton::get_thread_local_instance();
// Get a thread_local instance
```

**See also**

Code of qpp::Codes, qpp::Gates, qpp::Init, qpp::RandomDevices, qpp::States or qpp.h for real world examples of usage.

## 7.66.2 Constructor & Destructor Documentation

#### 7.66.2.1 Singleton() [1/2]

```
template<typename T>
qpp::internal::Singleton< T >::Singleton ( )  [protected], [default], [noexcept]
```

#### 7.66.2.2 Singleton() [2/2]

```
template<typename T>
qpp::internal::Singleton< T >::Singleton (
             const Singleton< T > &  )  [protected], [delete]
```

#### 7.66.2.3 ∼Singleton()

```
template<typename T>
virtual qpp::internal::Singleton< T >::∼Singleton ( )  [protected], [virtual], [default]
```

### 7.66.3 Member Function Documentation

#### 7.66.3.1 get_instance()

```
template<typename T>
static T& qpp::internal::Singleton< T >::get_instance ( )  [inline], [static], [noexcept]
```

#### 7.66.3.2 get_thread_local_instance()

```
template<typename T>
static T& qpp::internal::Singleton< T >::get_thread_local_instance ( )  [inline], [static],
[noexcept]
```

#### 7.66.3.3 operator=()

```
template<typename T>
Singleton& qpp::internal::Singleton< T >::operator= (
             const Singleton< T > &  )  [protected], [delete]
```

The documentation for this class was generated from the following file:

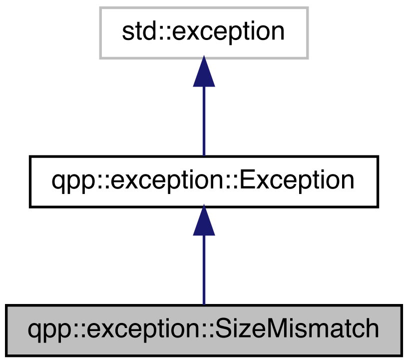
- internal/classes/singleton.h

## 7.67 qpp::exception::SizeMismatch Class Reference

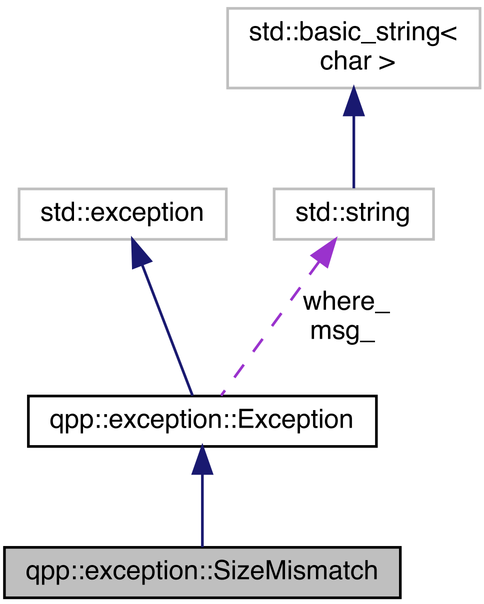
Size mismatch exception.

`#include <classes/exception.h>`

Inheritance diagram for qpp::exception::SizeMismatch:




Collaboration diagram for qpp::exception::SizeMismatch:



## Public Member Functions

- std::string type_description () const override
  *Exception type description.*
- Exception (const std::string &where)
  *Constructs an exception.*

### 7.67.1 Detailed Description

Size mismatch exception.

Sizes do not match

### 7.67.2 Member Function Documentation

#### 7.67.2.1 Exception()

`qpp::exception::Exception::Exception [inline], [explicit]`

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.67.2.2 type_description()

`std::string qpp::exception::SizeMismatch::type_description ( ) const [inline], [override], [virtual]`

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

- classes/exception.h

## 7.68 qpp::NoiseType::StateDependent Class Reference

Template tag, used whenever the noise is state-dependent.

`#include <classes/noise.h>`

### 7.68.1 Detailed Description

Template tag, used whenever the noise is state-dependent.

The documentation for this class was generated from the following file:

- classes/noise.h

## 7.69 qpp::NoiseType::StateIndependent Class Reference

Template tag, used whenever the noise is state-independent.

`#include <classes/noise.h>`

### 7.69.1 Detailed Description

Template tag, used whenever the noise is state-independent.

The documentation for this class was generated from the following file:

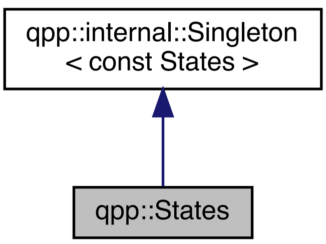
- classes/noise.h

## 7.70 qpp::States Class Reference

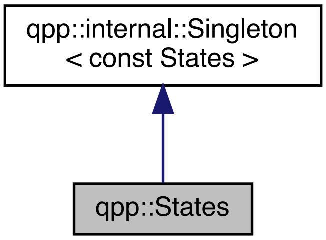
const Singleton class that implements most commonly used states

`#include <classes/states.h>`

Inheritance diagram for qpp::States:



Collaboration diagram for qpp::States:




## Public Member Functions

- ket mes (idx d=2) const

  *Maximally entangled state of 2 qudits.*
- ket zero (idx n, idx d=2) const

  *Zero state of n qudits.*
- ket one (idx n, idx d=2) const

  *One state of n qudits.*
- ket jn (idx j, idx n, idx d=2) const

  $|j\rangle^{\otimes n}$ *state of n qudits*
- ket plus (idx n) const

  *Plus state of n qubits.*
- ket minus (idx n) const

  *Minus state of n qubits.*

## Public Attributes

- ket x0 {ket::Zero(2)}

  *Pauli Sigma-X 0-eigenstate |+>*
- ket x1 {ket::Zero(2)}

  *Pauli Sigma-X 1-eigenstate |->*
- ket y0 {ket::Zero(2)}

  *Pauli Sigma-Y 0-eigenstate |y+>*
- ket y1 {ket::Zero(2)}

  *Pauli Sigma-Y 1-eigenstate |y->*
- ket z0 {ket::Zero(2)}

  *Pauli Sigma-Z 0-eigenstate |0>*
- ket z1 {ket::Zero(2)}

  *Pauli Sigma-Z 1-eigenstate |1>*
- cmat px0 {cmat::Zero(2, 2)}

  *Projector onto the Pauli Sigma-X 0-eigenstate |+><+|.*
- cmat px1 {cmat::Zero(2, 2)}

  *Projector onto the Pauli Sigma-X 1-eigenstate |-><-|.*

- cmat py0 {cmat::Zero(2, 2)}

  *Projector onto the Pauli Sigma-Y 0-eigenstate |y+><y+|.*
- cmat py1 {cmat::Zero(2, 2)}

  *Projector onto the Pauli Sigma-Y 1-eigenstate |y-><y-|.*
- cmat pz0 {cmat::Zero(2, 2)}

  *Projector onto the Pauli Sigma-Z 0-eigenstate |0><0|.*
- cmat pz1 {cmat::Zero(2, 2)}

  *Projector onto the Pauli Sigma-Z 1-eigenstate |1><1|.*
- ket b00 {ket::Zero(4)}

  *Bell-00 state, as described in Nielsen and Chuang.*
- ket b01 {ket::Zero(4)}

  *Bell-01 state, as described in Nielsen and Chuang.*
- ket b10 {ket::Zero(4)}

  *Bell-10 state, as described in Nielsen and Chuang.*
- ket b11 {ket::Zero(4)}

  *Bell-11 state, as described in Nielsen and Chuang.*
- cmat pb00 {cmat::Zero(4, 4)}

  *Projector onto the Bell-00 state.*
- cmat pb01 {cmat::Zero(4, 4)}

  *Projector onto the Bell-01 state.*
- cmat pb10 {cmat::Zero(4, 4)}

  *Projector onto the Bell-10 state.*
- cmat pb11 {cmat::Zero(4, 4)}

  *Projector onto the Bell-11 state.*
- ket GHZ {ket::Zero(8)}

  *GHZ state.*
- ket W {ket::Zero(8)}

  *W state.*
- cmat pGHZ {cmat::Zero(8, 8)}

  *Projector onto the GHZ state.*
- cmat pW {cmat::Zero(8, 8)}

  *Projector onto the W state.*

### Private Member Functions

- States ()
- ~States ()=default

  *Default destructor.*

### Friends

- class internal::Singleton< const States >

### Additional Inherited Members

## 7.70.1 Detailed Description

const Singleton class that implements most commonly used states

### 7.70.2 Constructor & Destructor Documentation

#### 7.70.2.1 States()

```
qpp::States::States ( )  [inline], [private]
```

Initialize the states

#### 7.70.2.2 ∼States()

```
qpp::States::~States ( )  [private], [default]
```

Default destructor.

### 7.70.3 Member Function Documentation

#### 7.70.3.1 jn()

```
ket qpp::States::jn (
            idx j,
            idx n,
            idx d = 2 ) const  [inline]
```

$|j\rangle^{\otimes n}$ state of *n* qudits

**Parameters**

| | |
|---|---|
| *j* | Non-negative integer |
| *n* | Non-negative integer |
| *d* | Subsystem dimensions |

**Returns**

$|j\rangle^{\otimes n}$ state of *n* qudits

#### 7.70.3.2 mes()

```
ket qpp::States::mes (
            idx d = 2 ) const  [inline]
```

Maximally entangled state of 2 qudits.

**Parameters**

| | |
|---|---|
| *d* | Subsystem dimensions |

**Returns**

Maximally entangled state $\frac{1}{\sqrt{d}}\sum_{j=0}^{d-1}|jj\rangle$ of 2 qudits

#### 7.70.3.3 minus()

```
ket qpp::States::minus (
            idx n ) const  [inline]
```

Minus state of *n* qubits.

**Parameters**

| | |
|---|---|
| *n* | Non-negative integer |

**Returns**

Minus state $|-\rangle^{\otimes n}$ of *n* qubits

#### 7.70.3.4 one()

```
ket qpp::States::one (
            idx n,
            idx d = 2 ) const  [inline]
```

One state of *n* qudits.

**Parameters**

| | |
|---|---|
| *n* | Non-negative integer |
| *d* | Subsystem dimensions |

**Returns**

One state $|1\rangle^{\otimes n}$ of *n* qudits

#### 7.70.3.5 plus()

```
ket qpp::States::plus (
            idx n ) const  [inline]
```

Plus state of *n* qubits.

**Parameters**

| | |
|---|---|
| *n* | Non-negative integer |

**Returns**

Plus state $|+\rangle^{\otimes n}$ of *n* qubits

#### 7.70.3.6 zero()

```
ket qpp::States::zero (
            idx n,
            idx d = 2 ) const  [inline]
```

Zero state of *n* qudits.

**Parameters**

| | |
|---|---|
| *n* | Non-negative integer |
| *d* | Subsystem dimensions |

**Returns**

Zero state $|0\rangle^{\otimes n}$ of *n* qudits

### 7.70.4 Friends And Related Function Documentation

#### 7.70.4.1 internal::Singleton< const States >

```
friend class internal::Singleton< const States >  [friend]
```

### 7.70.5 Member Data Documentation

#### 7.70.5.1 b00

```
ket qpp::States::b00 {ket::Zero(4)}
```

Bell-00 state, as described in Nielsen and Chuang.

#### 7.70.5.2 b01

```
ket qpp::States::b01 {ket::Zero(4)}
```

Bell-01 state, as described in Nielsen and Chuang.

#### 7.70.5.3 b10

```
ket qpp::States::b10 {ket::Zero(4)}
```

Bell-10 state, as described in Nielsen and Chuang.

#### 7.70.5.4 b11

```
ket qpp::States::b11 {ket::Zero(4)}
```

Bell-11 state, as described in Nielsen and Chuang.

#### 7.70.5.5 GHZ

```
ket qpp::States::GHZ {ket::Zero(8)}
```

GHZ state.

#### 7.70.5.6 pb00

```
cmat qpp::States::pb00 {cmat::Zero(4, 4)}
```

Projector onto the Bell-00 state.

#### 7.70.5.7 pb01

```
cmat qpp::States::pb01 {cmat::Zero(4, 4)}
```

Projector onto the Bell-01 state.

#### 7.70.5.8 pb10

```
cmat qpp::States::pb10 {cmat::Zero(4, 4)}
```

Projector onto the Bell-10 state.

#### 7.70.5.9 pb11

```
cmat qpp::States::pb11 {cmat::Zero(4, 4)}
```

Projector onto the Bell-11 state.

#### 7.70.5.10 pGHZ

```
cmat qpp::States::pGHZ {cmat::Zero(8, 8)}
```

Projector onto the GHZ state.

#### 7.70.5.11 pW

```
cmat qpp::States::pW {cmat::Zero(8, 8)}
```

Projector onto the W state.

#### 7.70.5.12 px0

```
cmat qpp::States::px0 {cmat::Zero(2, 2)}
```

Projector onto the Pauli Sigma-X 0-eigenstate $|+><+|$.

#### 7.70.5.13 px1

```
cmat qpp::States::px1 {cmat::Zero(2, 2)}
```

Projector onto the Pauli Sigma-X 1-eigenstate |-><-|.

#### 7.70.5.14 py0

```
cmat qpp::States::py0 {cmat::Zero(2, 2)}
```

Projector onto the Pauli Sigma-Y 0-eigenstate |y+><y+|.

#### 7.70.5.15 py1

```
cmat qpp::States::py1 {cmat::Zero(2, 2)}
```

Projector onto the Pauli Sigma-Y 1-eigenstate |y-><y-|.

#### 7.70.5.16 pz0

```
cmat qpp::States::pz0 {cmat::Zero(2, 2)}
```

Projector onto the Pauli Sigma-Z 0-eigenstate |0><0|.

#### 7.70.5.17 pz1

```
cmat qpp::States::pz1 {cmat::Zero(2, 2)}
```

Projector onto the Pauli Sigma-Z 1-eigenstate |1><1|.

#### 7.70.5.18 W

```
ket qpp::States::W {ket::Zero(8)}
```

W state.

#### 7.70.5.19 x0

```
ket qpp::States::x0 {ket::Zero(2)}
```

Pauli Sigma-X 0-eigenstate |+>

#### 7.70.5.20 x1

```
ket qpp::States::x1 {ket::Zero(2)}
```

Pauli Sigma-X 1-eigenstate |->

#### 7.70.5.21 y0

```
ket qpp::States::y0 {ket::Zero(2)}
```

Pauli Sigma-Y 0-eigenstate |y+>

#### 7.70.5.22 y1

```
ket qpp::States::y1 {ket::Zero(2)}
```

Pauli Sigma-Y 1-eigenstate |y->

#### 7.70.5.23 z0

```
ket qpp::States::z0 {ket::Zero(2)}
```

Pauli Sigma-Z 0-eigenstate |0>

#### 7.70.5.24 z1

```
ket qpp::States::z1 {ket::Zero(2)}
```

Pauli Sigma-Z 1-eigenstate |1>

The documentation for this class was generated from the following file:

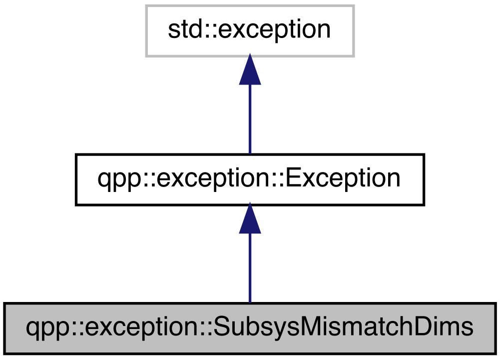
- classes/states.h

## 7.71 qpp::exception::SubsysMismatchDims Class Reference

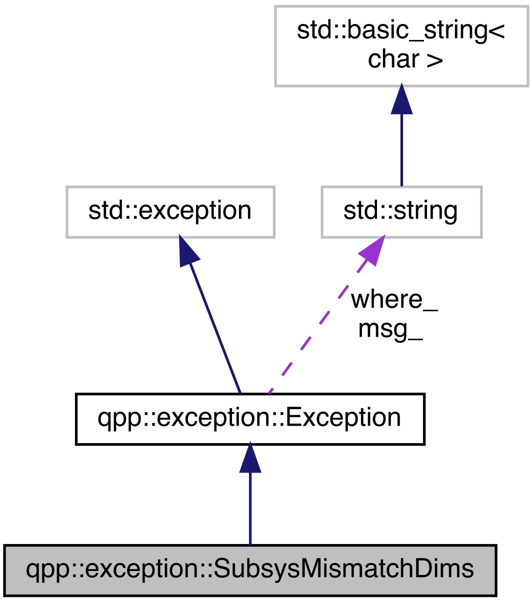
Subsystems mismatch dimensions exception.

`#include <classes/exception.h>`

Inheritance diagram for qpp::exception::SubsysMismatchDims:




Collaboration diagram for qpp::exception::SubsysMismatchDims:



## Public Member Functions

- std::string type_description () const override
  *Exception type description.*
- Exception (const std::string &where)
  *Constructs an exception.*

### 7.71.1 Detailed Description

Subsystems mismatch dimensions exception.

std::vector<idx> of subsystem labels has duplicates, or has entries that are larger than the size of the std↩ ::vector<idx> of dimensions

### 7.71.2 Member Function Documentation

#### 7.71.2.1 Exception()

```
qpp::exception::Exception::Exception  [inline], [explicit]
```

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.71.2.2 type_description()

```
std::string qpp::exception::SubsysMismatchDims::type_description ( ) const  [inline], [override],
[virtual]
```

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

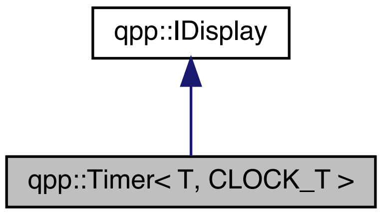
- classes/exception.h

## 7.72 qpp::Timer< T, CLOCK_T > Class Template Reference

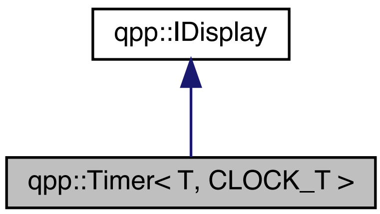
Chronometer.

```
#include <classes/timer.h>
```

Inheritance diagram for qpp::Timer< T, CLOCK_T >:




Collaboration diagram for qpp::Timer< T, CLOCK_T >:




### Public Member Functions

- Timer () noexcept

  *Constructs an instance with the current time as the starting point.*
- void tic () noexcept

  *Resets the chronometer.*
- const Timer & toc () noexcept

  *Stops the chronometer.*
- double tics () const noexcept

  *Time passed in the duration specified by T.*
- template<typename U = T>
  U get_duration () const noexcept

  *Duration specified by U.*
- Timer (const Timer &)=default

*Default copy constructor.*
- Timer (Timer &&)=default

  *Default move constructor.*
- Timer & operator= (const Timer &)=default

  *Default copy assignment operator.*
- Timer & operator= (Timer &&)=default

  *Default move assignment operator.*
- virtual ~Timer ()=default

  *Default virtual destructor.*

## Protected Attributes

- CLOCK_T::time_point start_
- CLOCK_T::time_point end_

## Private Member Functions

- std::ostream & display (std::ostream &os) const override

  *qpp::IDisplay::display() override*

## 7.72.1 Detailed Description

**template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady_clock>**
**class qpp::Timer< T, CLOCK_T >**

Chronometer.

**Template Parameters**

| | |
|---|---|
| *T* | Tics duration, default is std::chrono::duration<double, 1>, i.e. seconds in double precision |
| *CLOCK_T* | Clock's type, default is std::chrono::steady_clock, not affected by wall clock changes during runtime |

## 7.72.2 Constructor & Destructor Documentation

### 7.72.2.1 Timer() [1/3]

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady↩
_clock>
qpp::Timer< T, CLOCK_T >::Timer ( ) [inline], [noexcept]
```

Constructs an instance with the current time as the starting point.

#### 7.72.2.2 Timer() [2/3]

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady←
_clock>
qpp::Timer< T, CLOCK_T >::Timer (
            const Timer< T, CLOCK_T > & )  [default]
```

Default copy constructor.

#### 7.72.2.3 Timer() [3/3]

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady←
_clock>
qpp::Timer< T, CLOCK_T >::Timer (
            Timer< T, CLOCK_T > && )  [default]
```

Default move constructor.

#### 7.72.2.4 ∼Timer()

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady←
_clock>
virtual qpp::Timer< T, CLOCK_T >::∼Timer ( )  [virtual], [default]
```

Default virtual destructor.

### 7.72.3 Member Function Documentation

#### 7.72.3.1 display()

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady←
_clock>
std::ostream& qpp::Timer< T, CLOCK_T >::display (
            std::ostream & os ) const  [inline], [override], [private], [virtual]
```

qpp::IDisplay::display() override

Writes to the output stream the number of tics (specified by T) that passed between the instantiation/reset and invocation of qpp::Timer::toc().

**Parameters**

| | |
|---|---|
| *os* | Output stream passed by reference |

**Returns**

Reference to the output stream

Implements qpp::IDisplay.

#### 7.72.3.2 get_duration()

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady←
_clock>
template<typename U = T>
U qpp::Timer< T, CLOCK_T >::get_duration ( ) const  [inline], [noexcept]
```

Duration specified by U.

**Template Parameters**

| | |
|---|---|
| *U* | Duration, default is T, which defaults to std::chrono::duration<double, 1>, i.e. seconds in double precision |

**Returns**

Duration that passed between the instantiation/reset and invocation of qpp::Timer::toc()

#### 7.72.3.3 operator=() [1/2]

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady←
_clock>
Timer& qpp::Timer< T, CLOCK_T >::operator= (
            const Timer< T, CLOCK_T > & )  [default]
```

Default copy assignment operator.

#### 7.72.3.4 operator=() [2/2]

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady←
_clock>
Timer& qpp::Timer< T, CLOCK_T >::operator= (
            Timer< T, CLOCK_T > && )  [default]
```

Default move assignment operator.

#### 7.72.3.5 tic()

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady↩
_clock>
void qpp::Timer< T, CLOCK_T >::tic ( )  [inline], [noexcept]
```

Resets the chronometer.

Resets the starting/ending point to the current time

#### 7.72.3.6 tics()

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady↩
_clock>
double qpp::Timer< T, CLOCK_T >::tics ( ) const  [inline], [noexcept]
```

Time passed in the duration specified by T.

**Returns**

Number of tics (specified by T) that passed between the instantiation/reset and invocation of qpp::Timer::toc()

#### 7.72.3.7 toc()

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady↩
_clock>
const Timer& qpp::Timer< T, CLOCK_T >::toc ( )  [inline], [noexcept]
```

Stops the chronometer.

Set the current time as the ending point

**Returns**

Reference to the current instance

### 7.72.4 Member Data Documentation

#### 7.72.4.1 end_

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady↩
_clock>
CLOCK_T::time_point qpp::Timer< T, CLOCK_T >::end_  [protected]
```

#### 7.72.4.2 start_

```
template<typename T = std::chrono::duration<double>, typename CLOCK_T = std::chrono::steady↩
_clock>
CLOCK_T::time_point qpp::Timer< T, CLOCK_T >::start_ [protected]
```

The documentation for this class was generated from the following file:

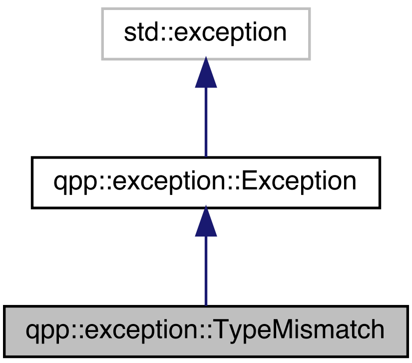
- classes/timer.h

## 7.73 qpp::exception::TypeMismatch Class Reference

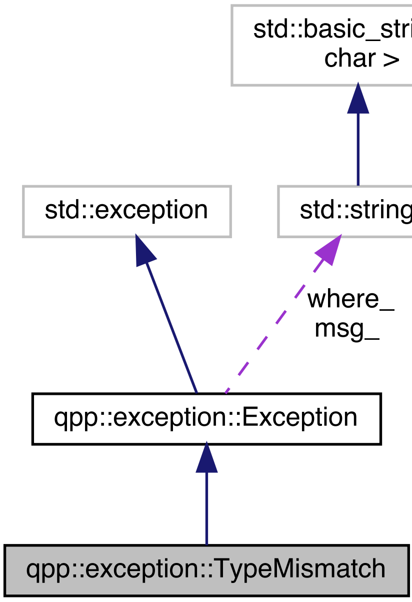
Type mismatch exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::TypeMismatch:



Collaboration diagram for qpp::exception::TypeMismatch:




## Public Member Functions

- std::string type_description () const override
  *Exception type description.*
- Exception (const std::string &where)
  *Constructs an exception.*

### 7.73.1 Detailed Description

Type mismatch exception.

Scalar types do not match

### 7.73.2 Member Function Documentation

#### 7.73.2.1 Exception()

```
qpp::exception::Exception::Exception  [inline], [explicit]
```

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.73.2.2 type_description()

```
std::string qpp::exception::TypeMismatch::type_description ( ) const  [inline], [override],
[virtual]
```

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

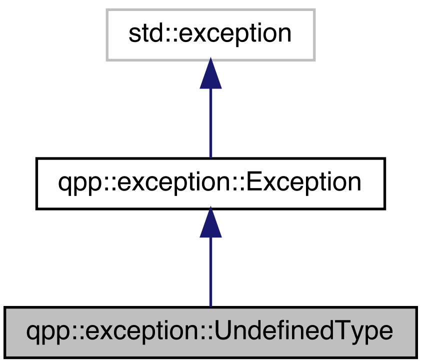
- classes/exception.h

## 7.74 qpp::exception::UndefinedType Class Reference

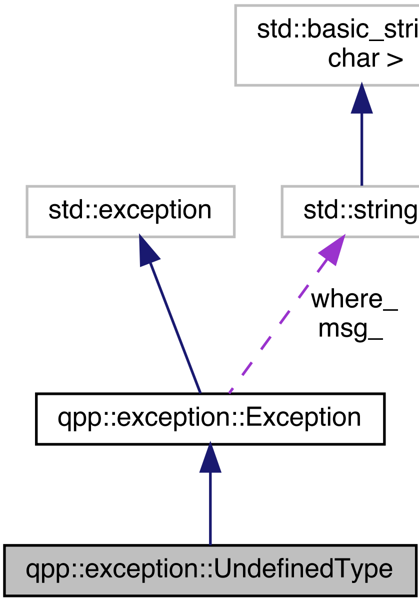
Not defined for this type exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::UndefinedType:



Collaboration diagram for qpp::exception::UndefinedType:




### Public Member Functions

- std::string type_description () const override

  *Exception type description.*
- Exception (const std::string &where)

  *Constructs an exception.*

## 7.74.1 Detailed Description

Not defined for this type exception.

Templated specialization is not defined for this type

## 7.74.2 Member Function Documentation

#### 7.74.2.1 Exception()

```
qpp::exception::Exception::Exception  [inline], [explicit]
```

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.74.2.2 type_description()

```
std::string qpp::exception::UndefinedType::type_description ( ) const  [inline], [override],
[virtual]
```

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

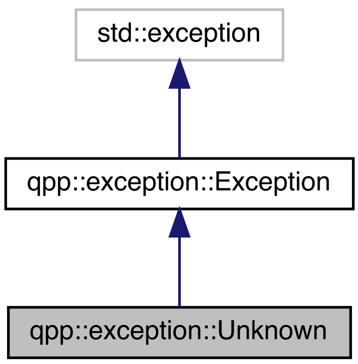
- classes/exception.h

## 7.75 qpp::exception::Unknown Class Reference

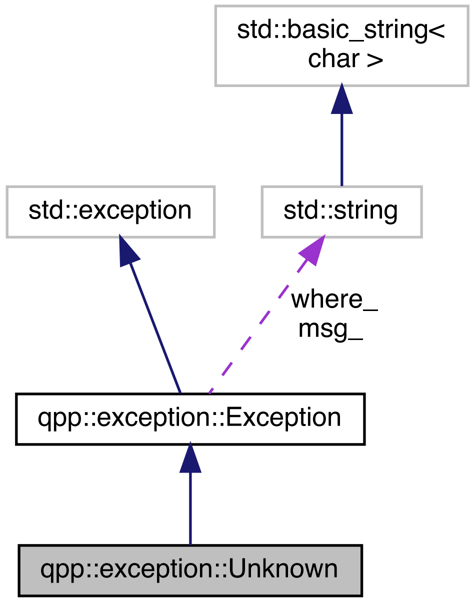
Unknown exception.

```
#include <classes/exception.h>
```

Inheritance diagram for qpp::exception::Unknown:



Collaboration diagram for qpp::exception::Unknown:




## Public Member Functions

- std::string type_description () const override
  *Exception type description.*
- Exception (const std::string &where)
  *Constructs an exception.*

### 7.75.1 Detailed Description

Unknown exception.

Thrown when no other exception is suitable (not recommended, it is better to define another suitable exception type)

### 7.75.2 Member Function Documentation

#### 7.75.2.1 Exception()

```
qpp::exception::Exception::Exception  [inline], [explicit]
```

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

#### 7.75.2.2 type_description()

```
std::string qpp::exception::Unknown::type_description ( ) const  [inline], [override], [virtual]
```

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

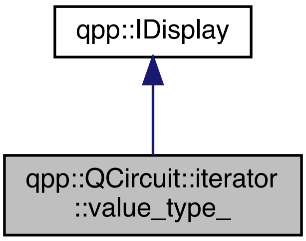
The documentation for this class was generated from the following file:

- classes/exception.h

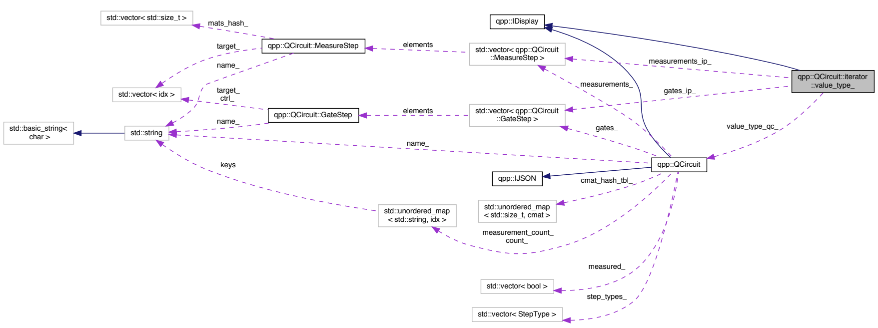
## 7.76 qpp::QCircuit::iterator::value_type_ Class Reference

Value type class for qpp::QCircuit::iterator.

Inheritance diagram for qpp::QCircuit::iterator::value_type_:




Collaboration diagram for qpp::QCircuit::iterator::value_type_:



### Public Member Functions

- value_type_ (const QCircuit ∗value_type_qc)

  *Default value_type_ constructor.*
- value_type_ (const value_type_ &)=default

  *Default copy constructor.*
- value_type_ & operator= (const value_type_ &)=default

  *Default copy assignment operator.*

### Public Attributes

- const QCircuit ∗ value_type_qc_

  *< non-owning pointer to the grand-parent const quantum circuit*
- StepType type_ {StepType::NONE}

  *step type*
- idx ip_ {static_cast<idx>(-1)}

  *instruction pointer*
- std::vector< GateStep >::const_iterator gates_ip_ {}

  *gates instruction pointer*
- std::vector< MeasureStep >::const_iterator measurements_ip_ {}

  *measurements instruction pointer*

### Private Member Functions

- std::ostream & display (std::ostream &os) const override

  *qpp::IDisplay::display() override*

## 7.76.1 Detailed Description

Value type class for qpp::QCircuit::iterator.

## 7.76.2 Constructor & Destructor Documentation

#### 7.76.2.1 value_type_() [1/2]

```
qpp::QCircuit::iterator::value_type_::value_type_ (
            const QCircuit * value_type_qc )  [inline], [explicit]
```

Default value_type_ constructor.

**Parameters**

| | |
|---|---|
| *value_type_qc* | Pointer to constant quantum circuit |

#### 7.76.2.2 value_type_() [2/2]

```
qpp::QCircuit::iterator::value_type_::value_type_ (
            const value_type_ & ) [default]
```

Default copy constructor.

### 7.76.3 Member Function Documentation

#### 7.76.3.1 display()

```
std::ostream& qpp::QCircuit::iterator::value_type_::display (
            std::ostream & os ) const [inline], [override], [private], [virtual]
```

qpp::IDisplay::display() override

Writes to the output stream the textual representation of the iterator de-referenced element

**Parameters**

| | |
|---|---|
| *os* | Output stream passed by reference |

**Returns**

Reference to the output stream

Implements qpp::IDisplay.

#### 7.76.3.2 operator=()

```
value_type_& qpp::QCircuit::iterator::value_type_::operator= (
            const value_type_ & ) [default]
```

Default copy assignment operator.

**Returns**

Reference to the current instance

### 7.76.4 Member Data Documentation

#### 7.76.4.1 gates_ip_

```
std::vector<GateStep>::const_iterator qpp::QCircuit::iterator::value_type_::gates_ip_ {}
```

gates instruction pointer

#### 7.76.4.2 ip_

```
idx qpp::QCircuit::iterator::value_type_::ip_ {static_cast<idx>(-1)}
```

instruction pointer

#### 7.76.4.3 measurements_ip_

```
std::vector<MeasureStep>::const_iterator qpp::QCircuit::iterator::value_type_::measurements_←↩
ip_ {}
```

measurements instruction pointer

#### 7.76.4.4 type_

```
StepType qpp::QCircuit::iterator::value_type_::type_ {StepType::NONE}
```

step type

#### 7.76.4.5 value_type_qc_

```
const QCircuit* qpp::QCircuit::iterator::value_type_::value_type_qc_
```

$<$ non-owning pointer to the grand-parent const quantum circuit

The documentation for this class was generated from the following file:

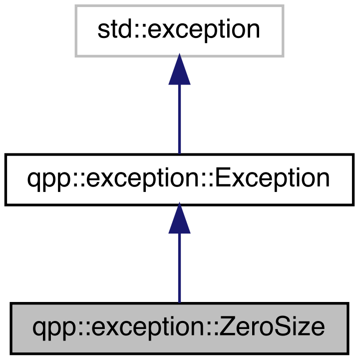
- classes/circuits/circuits.h

## 7.77 qpp::exception::ZeroSize Class Reference

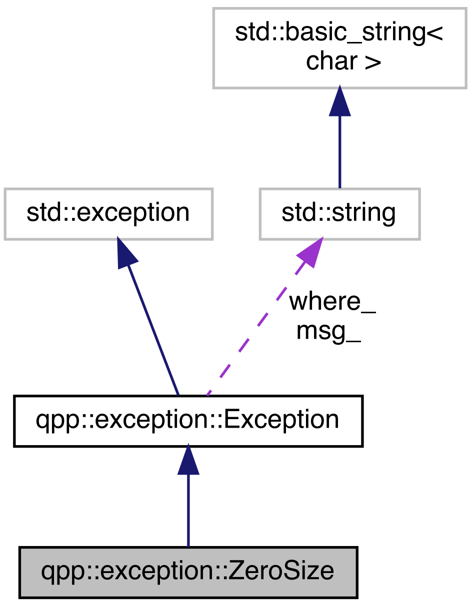
Object has zero size exception.

`#include <classes/exception.h>`

Inheritance diagram for qpp::exception::ZeroSize:




Collaboration diagram for qpp::exception::ZeroSize:



### Public Member Functions

- std::string type_description () const override

  *Exception type description.*
- Exception (const std::string &where)

  *Constructs an exception.*

## 7.77.1 Detailed Description

Object has zero size exception.

Zero sized object, e.g. empty Eigen::Matrix or std::vector with no elements

## 7.77.2 Member Function Documentation

### 7.77.2.1 Exception()

```
qpp::exception::Exception::Exception  [inline], [explicit]
```

Constructs an exception.

**Parameters**

| | |
|---|---|
| *where* | Text representing where the exception occurred |

### 7.77.2.2 type_description()

```
std::string qpp::exception::ZeroSize::type_description ( ) const  [inline], [override], [virtual]
```

Exception type description.

**Returns**

Exception type description

Implements qpp::exception::Exception.

The documentation for this class was generated from the following file:

- classes/exception.h

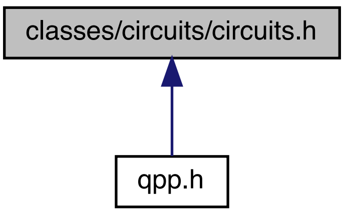
# Chapter 8

# File Documentation

## 8.1 classes/circuits/circuits.h File Reference

Qudit quantum circuits.

This graph shows which files directly or indirectly include this file:




### Classes

- class qpp::QCircuit

  *Quantum circuit class.*
- struct qpp::QCircuit::GateStep

  *One step consisting only of gates/operators in the circuit.*
- struct qpp::QCircuit::MeasureStep

  *One step consisting only of measurements in the circuit.*
- class qpp::QCircuit::iterator

  *Quantum circuit bound-checking (safe) iterator.*
- class qpp::QCircuit::iterator::value_type_

  *Value type class for qpp::QCircuit::iterator.*

## Namespaces

- qpp

    *Quantum++ main namespace.*

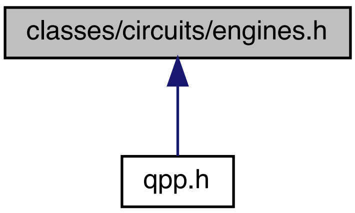
### 8.1.1 Detailed Description

Qudit quantum circuits.

## 8.2 classes/circuits/engines.h File Reference

Qudit quantum engines.

This graph shows which files directly or indirectly include this file:




## Classes

- class qpp::QEngine

    *Quantum circuit engine, executes qpp::QCircuit.*
- class qpp::QNoisyEngine< NoiseModel >

    *Noisy quantum circuit engine, executes qpp::QCircuit.*

## Namespaces

- qpp

    *Quantum++ main namespace.*

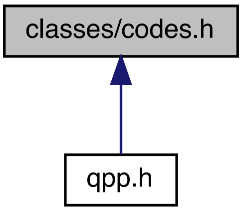
### 8.2.1 Detailed Description

Qudit quantum engines.

## 8.3 classes/codes.h File Reference

Quantum error correcting codes.

This graph shows which files directly or indirectly include this file:




### Classes

- class qpp::Codes

  *const Singleton class that defines quantum error correcting codes*

### Namespaces

- qpp

  *Quantum++ main namespace.*

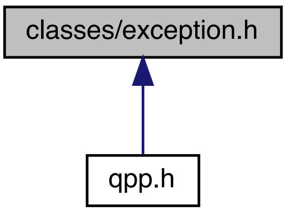
### 8.3.1 Detailed Description

Quantum error correcting codes.

## 8.4 classes/exception.h File Reference

Exceptions.

This graph shows which files directly or indirectly include this file:



## Classes

- class qpp::exception::Exception

  *Base class for generating Quantum++ custom exceptions.*
- class qpp::exception::Unknown

  *Unknown exception.*
- class qpp::exception::ZeroSize

  *Object has zero size exception.*
- class qpp::exception::MatrixNotSquare

  *Matrix is not square exception.*
- class qpp::exception::MatrixNotCvector

  *Matrix is not a column vector exception.*
- class qpp::exception::MatrixNotRvector

  *Matrix is not a row vector exception.*
- class qpp::exception::MatrixNotVector

  *Matrix is not a vector exception.*
- class qpp::exception::MatrixNotSquareNorCvector

  *Matrix is not square nor column vector exception.*
- class qpp::exception::MatrixNotSquareNorRvector

  *Matrix is not square nor row vector exception.*
- class qpp::exception::MatrixNotSquareNorVector

  *Matrix is not square nor vector exception.*
- class qpp::exception::MatrixMismatchSubsys

  *Matrix mismatch subsystems exception.*
- class qpp::exception::DimsInvalid

  *Invalid dimension(s) exception.*
- class qpp::exception::DimsNotEqual

  *Dimensions not equal exception.*
- class qpp::exception::DimsMismatchMatrix

  *Dimension(s) mismatch matrix size exception.*
- class qpp::exception::DimsMismatchCvector

  *Dimension(s) mismatch column vector size exception.*
- class qpp::exception::DimsMismatchRvector

  *Dimension(s) mismatch row vector size exception.*
- class qpp::exception::DimsMismatchVector

  *Dimension(s) mismatch vector size exception.*
- class qpp::exception::SubsysMismatchDims

  *Subsystems mismatch dimensions exception.*
- class qpp::exception::PermInvalid

  *Invalid permutation exception.*
- class qpp::exception::PermMismatchDims

  *Permutation mismatch dimensions exception.*
- class qpp::exception::NotQubitMatrix

  *Matrix is not 2 x 2 exception.*
- class qpp::exception::NotQubitCvector

  *Column vector is not 2 x 1 exception.*
- class qpp::exception::NotQubitRvector

  *Row vector is not 1 x 2 exception.*
- class qpp::exception::NotQubitVector

  *Vector is not 2 x 1 nor 1 x 2 exception.*
- class qpp::exception::NotQubitSubsys

*Subsystems are not qubits exception.*
- class qpp::exception::NotBipartite

  *Not bi-partite exception.*
- class qpp::exception::NoCodeword

  *Codeword does not exist exception.*
- class qpp::exception::OutOfRange

  *Argument out of range exception.*
- class qpp::exception::TypeMismatch

  *Type mismatch exception.*
- class qpp::exception::SizeMismatch

  *Size mismatch exception.*
- class qpp::exception::UndefinedType

  *Not defined for this type exception.*
- class qpp::exception::QuditAlreadyMeasured

  *Qudit was already measured exception.*
- class qpp::exception::Duplicates

  *System (e.g. std::vector) has duplicates exception.*
- class qpp::exception::CustomException

  *Custom exception.*
- class qpp::exception::NotImplemented

  *Code not yet implemented.*
- class qpp::exception::InvalidIterator

  *Invalid iterator.*

### Namespaces

- qpp

  *Quantum++ main namespace.*
- qpp::exception

  *Quantum++ exception hierarchy namespace.*

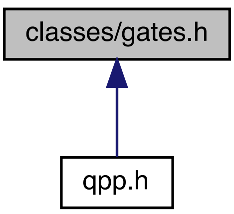
### 8.4.1 Detailed Description

Exceptions.

## 8.5 classes/gates.h File Reference

Quantum gates.

This graph shows which files directly or indirectly include this file:



## Classes

- class qpp::Gates

    *const Singleton class that implements most commonly used gates*

## Namespaces

- qpp

    *Quantum++ main namespace.*

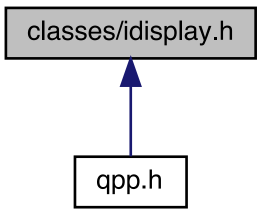
### 8.5.1 Detailed Description

Quantum gates.

## 8.6 classes/idisplay.h File Reference

Display interface via the non-virtual interface (NVI) and very basic JSON serialization support interface.

This graph shows which files directly or indirectly include this file:




## Classes

- class qpp::IDisplay

    *Abstract class (interface) that mandates the definition of virtual std::ostream& display(std::ostream& os) const.*
- class qpp::IJSON

    *Abstract class (interface) that mandates the definition of very basic JSON serialization support.*

## Namespaces

- qpp

    *Quantum++ main namespace.*

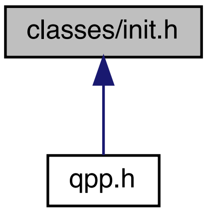
### 8.6.1 Detailed Description

Display interface via the non-virtual interface (NVI) and very basic JSON serialization support interface.

## 8.7 classes/init.h File Reference

Initialization.

This graph shows which files directly or indirectly include this file:




### Classes

- class qpp::Init

    *const Singleton class that performs additional initializations/cleanups*

### Namespaces

- qpp

    *Quantum++ main namespace.*

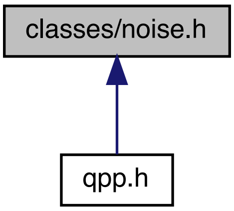
### 8.7.1 Detailed Description

Initialization.

## 8.8 classes/noise.h File Reference

Noise models.

This graph shows which files directly or indirectly include this file:




### Classes

- class qpp::NoiseType

    *Contains template tags used to specify the noise type.*
- class qpp::NoiseBase$<$ T $>$

    *Base class for all noise models, derive your particular noise model.*
- class qpp::QubitDepolarizingNoise

    *Qubit depolarizing noise.*
- class qpp::QubitPhaseFlipNoise

    *Qubit phase flip (dephasing) noise.*
- class qpp::QubitBitFlipNoise

    *Qubit bit flip noise.*
- class qpp::QubitBitPhaseFlipNoise

    *Qubit bit-phase flip (dephasing) noise.*
- class qpp::QubitAmplitudeDampingNoise

    *Qubit amplitude damping noise, as described in Nielsen and Chuang.*
- class qpp::QubitPhaseDampingNoise

    *Qubit phase damping noise, as described in Nielsen and Chuang.*
- class qpp::QuditDepolarizingNoise

    *Qudit depolarizing noise.*

### Namespaces

- qpp

    *Quantum++ main namespace.*

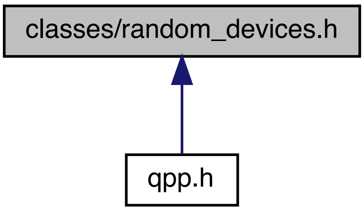
### 8.8.1 Detailed Description

Noise models.

## 8.9 classes/random_devices.h File Reference

Random devices.

This graph shows which files directly or indirectly include this file:




### Classes

- class qpp::RandomDevices

  *Singleton class that manages the source of randomness in the library.*

### Namespaces

- qpp

  *Quantum++ main namespace.*

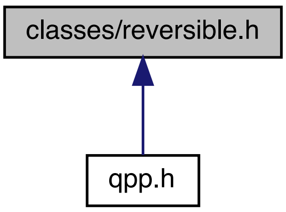
### 8.9.1 Detailed Description

Random devices.

## 8.10 classes/reversible.h File Reference

Support for classical reversible circuits.

This graph shows which files directly or indirectly include this file:



### Classes

- class qpp::Dynamic_bitset

    *Dynamic bitset class, allows the specification of the number of bits at runtime.*
- class qpp::Bit_circuit

    *Classical reversible circuit simulator.*

### Namespaces

- qpp

    *Quantum++ main namespace.*

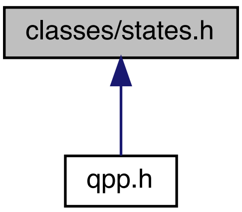
### 8.10.1 Detailed Description

Support for classical reversible circuits.

## 8.11 classes/states.h File Reference

Quantum states.

This graph shows which files directly or indirectly include this file:




### Classes

- class qpp::States

    *const Singleton class that implements most commonly used states*

### Namespaces

- qpp

    *Quantum++ main namespace.*

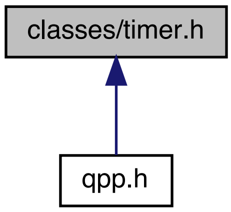
### 8.11.1 Detailed Description

Quantum states.

## 8.12 classes/timer.h File Reference

Timing.

This graph shows which files directly or indirectly include this file:




### Classes

- class qpp::Timer< T, CLOCK_T >

  *Chronometer.*

### Namespaces

- qpp

  *Quantum++ main namespace.*

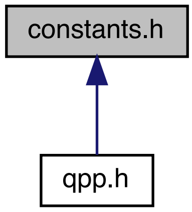
### 8.12.1 Detailed Description

Timing.

## 8.13 constants.h File Reference

Constants.

This graph shows which files directly or indirectly include this file:




### Namespaces

- qpp

    *Quantum++ main namespace.*
- qpp::literals

### Functions

- constexpr cplx qpp::literals::operator"" _i (unsigned long long int x) noexcept

    *User-defined literal for complex* $i = \sqrt{-1}$ *(integer overload)*
- constexpr cplx qpp::operator"" _i (long double x) noexcept

    *User-defined literal for complex* $i = \sqrt{-1}$ *(real overload)*
- cplx qpp::omega (idx D)

    *D-th root of unity.*

### Variables

- constexpr double qpp::chop = 1e-10

    *Used in qpp::disp() for setting to zero numbers that have their absolute value smaller than qpp::chop.*
- constexpr idx qpp::maxn = 64

    *Maximum number of allowed qubits/qudits (subsystems)*
- constexpr double qpp::pi = 3.141592653589793238462643383279502884

    $\pi$
- constexpr double qpp::ee = 2.718281828459045235360287471352662497

    *Base of natural logarithm,* $e$*.*
- constexpr double qpp::infty = std::numeric_limits<double>::max()

    *Used to denote infinity in double precision.*

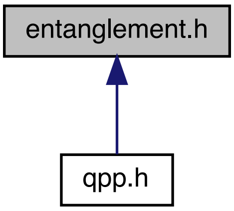
### 8.13.1 Detailed Description

Constants.

## 8.14 entanglement.h File Reference

Entanglement functions.

This graph shows which files directly or indirectly include this file:




### Namespaces

- qpp
  *Quantum++ main namespace.*

### Functions

- template< typename Derived >
  dyn_col_vect< double > qpp::schmidtcoeffs (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)
  *Schmidt coefficients of the bi-partite pure state A.*
- template< typename Derived >
  dyn_col_vect< double > qpp::schmidtcoeffs (const Eigen::MatrixBase< Derived > &A, idx d=2)
  *Schmidt coefficients of the bi-partite pure state A.*
- template< typename Derived >
  cmat qpp::schmidtA (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)
  *Schmidt basis on Alice side.*
- template< typename Derived >
  cmat qpp::schmidtA (const Eigen::MatrixBase< Derived > &A, idx d=2)
  *Schmidt basis on Alice side.*
- template< typename Derived >
  cmat qpp::schmidtB (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)
  *Schmidt basis on Bob side.*
- template< typename Derived >
  cmat qpp::schmidtB (const Eigen::MatrixBase< Derived > &A, idx d=2)

*Schmidt basis on Bob side.*

- template<typename Derived >
  std::vector< double > qpp::schmidtprobs (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)

  *Schmidt probabilities of the bi-partite pure state A.*

- template<typename Derived >
  std::vector< double > qpp::schmidtprobs (const Eigen::MatrixBase< Derived > &A, idx d=2)

  *Schmidt probabilities of the bi-partite pure state A.*

- template<typename Derived >
  double qpp::entanglement (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)

  *Entanglement of the bi-partite pure state A.*

- template<typename Derived >
  double qpp::entanglement (const Eigen::MatrixBase< Derived > &A, idx d=2)

  *Entanglement of the bi-partite pure state A.*

- template<typename Derived >
  double qpp::gconcurrence (const Eigen::MatrixBase< Derived > &A)

  *G-concurrence of the bi-partite pure state A.*

- template<typename Derived >
  double qpp::negativity (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)

  *Negativity of the bi-partite mixed state A.*

- template<typename Derived >
  double qpp::negativity (const Eigen::MatrixBase< Derived > &A, idx d=2)

  *Negativity of the bi-partite mixed state A.*

- template<typename Derived >
  double qpp::lognegativity (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)

  *Logarithmic negativity of the bi-partite mixed state A.*

- template<typename Derived >
  double qpp::lognegativity (const Eigen::MatrixBase< Derived > &A, idx d=2)

  *Logarithmic negativity of the bi-partite mixed state A.*

- template<typename Derived >
  double qpp::concurrence (const Eigen::MatrixBase< Derived > &A)

  *Wootters concurrence of the bi-partite qubit mixed state A.*

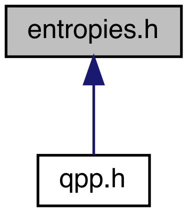
### 8.14.1 Detailed Description

Entanglement functions.

## 8.15 entropies.h File Reference

Entropy functions.

This graph shows which files directly or indirectly include this file:




## Namespaces

- qpp

    *Quantum++ main namespace.*

## Functions

- template< typename Derived >
  double qpp::entropy (const Eigen::MatrixBase< Derived > &A)

    *von-Neumann entropy of the density matrix A*
- double qpp::entropy (const std::vector< double > &prob)

    *Shannon entropy of the probability distribution prob.*
- template< typename Derived >
  double qpp::renyi (const Eigen::MatrixBase< Derived > &A, double alpha)

    *Renyi- $\alpha$ entropy of the density matrix A, for $\alpha \geq 0$.*
- double qpp::renyi (const std::vector< double > &prob, double alpha)

    *Renyi- $\alpha$ entropy of the probability distribution prob, for $\alpha \geq 0$.*
- template< typename Derived >
  double qpp::tsallis (const Eigen::MatrixBase< Derived > &A, double q)

    *Tsallis- $q$ entropy of the density matrix A, for $q \geq 0$.*
- double qpp::tsallis (const std::vector< double > &prob, double q)

    *Tsallis- $q$ entropy of the probability distribution prob, for $q \geq 0$.*
- template< typename Derived >
  double qpp::qmutualinfo (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &subsysA, const std::vector< idx > &subsysB, const std::vector< idx > &dims)

    *Quantum mutual information between 2 subsystems of a composite system.*
- template< typename Derived >
  double qpp::qmutualinfo (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &subsysA, const std::vector< idx > &subsysB, idx d=2)

    *Quantum mutual information between 2 subsystems of a composite system.*

### 8.15.1 Detailed Description

Entropy functions.

## 8.16 experimental/experimental.h File Reference

Experimental/test functions/classes.

### Namespaces

- qpp

    *Quantum++ main namespace.*
- qpp::experimental

    *Experimental/test functions/classes, do not use or modify.*

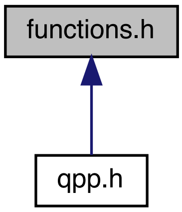
### 8.16.1 Detailed Description

Experimental/test functions/classes.

## 8.17 functions.h File Reference

Generic quantum computing functions.

This graph shows which files directly or indirectly include this file:




### Classes

- class qpp::internal::HashEigen

    *Functor for hashing Eigen expressions.*
- class qpp::internal::EqualEigen

    *Functor for comparing Eigen expressions for equality.*

### Namespaces

- qpp

    *Quantum++ main namespace.*
- qpp::literals
- qpp::internal

    *Internal utility functions, do not use them directly or modify them.*

## Functions

- template< typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::transpose (const Eigen::MatrixBase< Derived > &A)
  *Transpose.*
- template< typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::conjugate (const Eigen::MatrixBase< Derived > &A)
  *Complex conjugate.*
- template< typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::adjoint (const Eigen::MatrixBase< Derived > &A)
  *Adjoint.*
- template< typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::inverse (const Eigen::MatrixBase< Derived > &A)
  *Inverse.*
- template< typename Derived >
  Derived::Scalar qpp::trace (const Eigen::MatrixBase< Derived > &A)
  *Trace.*
- template< typename Derived >
  Derived::Scalar qpp::det (const Eigen::MatrixBase< Derived > &A)
  *Determinant.*
- template< typename Derived >
  Derived::Scalar qpp::logdet (const Eigen::MatrixBase< Derived > &A)
  *Logarithm of the determinant.*
- template< typename Derived >
  Derived::Scalar qpp::sum (const Eigen::MatrixBase< Derived > &A)
  *Element-wise sum of A.*
- template< typename Derived >
  Derived::Scalar qpp::prod (const Eigen::MatrixBase< Derived > &A)
  *Element-wise product of A.*
- template< typename Derived >
  double qpp::norm (const Eigen::MatrixBase< Derived > &A)
  *Frobenius norm.*
- template< typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::normalize (const Eigen::MatrixBase< Derived > &A)
  *Normalizes state vector (column or row vector) or density matrix.*
- template< typename Derived >
  std::pair< dyn_col_vect< cplx >, cmat > qpp::eig (const Eigen::MatrixBase< Derived > &A)
  *Full eigen decomposition.*
- template< typename Derived >
  dyn_col_vect< cplx > qpp::evals (const Eigen::MatrixBase< Derived > &A)
  *Eigenvalues.*
- template< typename Derived >
  cmat qpp::evects (const Eigen::MatrixBase< Derived > &A)
  *Eigenvectors.*
- template< typename Derived >
  std::pair< dyn_col_vect< double >, cmat > qpp::heig (const Eigen::MatrixBase< Derived > &A)
  *Full eigen decomposition of Hermitian expression.*
- template< typename Derived >
  dyn_col_vect< double > qpp::hevals (const Eigen::MatrixBase< Derived > &A)
  *Hermitian eigenvalues.*
- template< typename Derived >
  cmat qpp::hevects (const Eigen::MatrixBase< Derived > &A)
  *Eigenvectors of Hermitian matrix.*

- template<typename Derived >
std::tuple< cmat, dyn_col_vect< double >, cmat > qpp::svd (const Eigen::MatrixBase< Derived > &A)

    *Full singular value decomposition.*
- template<typename Derived >
dyn_col_vect< double > qpp::svals (const Eigen::MatrixBase< Derived > &A)

    *Singular values.*
- template<typename Derived >
cmat qpp::svdU (const Eigen::MatrixBase< Derived > &A)

    *Left singular vectors.*
- template<typename Derived >
cmat qpp::svdV (const Eigen::MatrixBase< Derived > &A)

    *Right singular vectors.*
- template<typename Derived >
cmat qpp::funm (const Eigen::MatrixBase< Derived > &A, cplx(∗f)(const cplx &))

    *Functional calculus f(A)*
- template<typename Derived >
cmat qpp::sqrtm (const Eigen::MatrixBase< Derived > &A)

    *Matrix square root.*
- template<typename Derived >
cmat qpp::absm (const Eigen::MatrixBase< Derived > &A)

    *Matrix absolute value.*
- template<typename Derived >
cmat qpp::expm (const Eigen::MatrixBase< Derived > &A)

    *Matrix exponential.*
- template<typename Derived >
cmat qpp::logm (const Eigen::MatrixBase< Derived > &A)

    *Matrix logarithm.*
- template<typename Derived >
cmat qpp::sinm (const Eigen::MatrixBase< Derived > &A)

    *Matrix sin.*
- template<typename Derived >
cmat qpp::cosm (const Eigen::MatrixBase< Derived > &A)

    *Matrix cos.*
- template<typename Derived >
cmat qpp::spectralpowm (const Eigen::MatrixBase< Derived > &A, const cplx z)

    *Matrix power.*
- template<typename Derived >
dyn_mat< typename Derived::Scalar > qpp::powm (const Eigen::MatrixBase< Derived > &A, idx n)

    *Fast matrix power based on the SQUARE-AND-MULTIPLY algorithm.*
- template<typename Derived >
double qpp::schatten (const Eigen::MatrixBase< Derived > &A, double p)

    *Schatten matrix norm.*
- template<typename OutputScalar , typename Derived >
dyn_mat< OutputScalar > qpp::cwise (const Eigen::MatrixBase< Derived > &A, OutputScalar(∗f)(const typename Derived::Scalar &))

    *Functor.*
- template<typename T >
dyn_mat< typename T::Scalar > qpp::kron (const T &head)

    *Kronecker product.*
- template<typename T , typename... Args>
dyn_mat< typename T::Scalar > qpp::kron (const T &head, const Args &... tail)

    *Kronecker product.*

- template<typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::kron (const std::vector< Derived > &As)

  *Kronecker product.*
- template<typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::kron (const std::initializer_list< Derived > &As)

  *Kronecker product.*
- template<typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::kronpow (const Eigen::MatrixBase< Derived > &A, idx n)

  *Kronecker power.*
- template<typename T >
  dyn_mat< typename T::Scalar > qpp::dirsum (const T &head)

  *Direct sum.*
- template<typename T , typename... Args>
  dyn_mat< typename T::Scalar > qpp::dirsum (const T &head, const Args &... tail)

  *Direct sum.*
- template<typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::dirsum (const std::vector< Derived > &As)

  *Direct sum.*
- template<typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::dirsum (const std::initializer_list< Derived > &As)

  *Direct sum.*
- template<typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::dirsumpow (const Eigen::MatrixBase< Derived > &A, idx n)

  *Direct sum power.*
- template<typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::reshape (const Eigen::MatrixBase< Derived > &A, idx rows, idx cols)

  *Reshape.*
- template<typename Derived1 , typename Derived2 >
  dyn_mat< typename Derived1::Scalar > qpp::comm (const Eigen::MatrixBase< Derived1 > &A, const Eigen::MatrixBase< Derived2 > &B)

  *Commutator.*
- template<typename Derived1 , typename Derived2 >
  dyn_mat< typename Derived1::Scalar > qpp::anticomm (const Eigen::MatrixBase< Derived1 > &A, const Eigen::MatrixBase< Derived2 > &B)

  *Anti-commutator.*
- template<typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::prj (const Eigen::MatrixBase< Derived > &A)

  *Projector.*
- template<typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::grams (const std::vector< Derived > &As)

  *Gram-Schmidt orthogonalization.*
- template<typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::grams (const std::initializer_list< Derived > &As)

  *Gram-Schmidt orthogonalization.*
- template<typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::grams (const Eigen::MatrixBase< Derived > &A)

  *Gram-Schmidt orthogonalization.*
- std::vector< idx > qpp::n2multiidx (idx n, const std::vector< idx > &dims)

  *Non-negative integer index to multi-index.*
- idx qpp::multiidx2n (const std::vector< idx > &midx, const std::vector< idx > &dims)

  *Multi-index to non-negative integer index.*
- ket qpp::mket (const std::vector< idx > &mask, const std::vector< idx > &dims)

*Multi-partite qudit ket.*
- ket qpp::mket (const std::vector< idx > &mask, idx d=2)

  *Multi-partite qudit ket.*
- cmat qpp::mprj (const std::vector< idx > &mask, const std::vector< idx > &dims)

  *Projector onto multi-partite qudit ket.*
- cmat qpp::mprj (const std::vector< idx > &mask, idx d=2)

  *Projector onto multi-partite qudit ket.*
- template<typename InputIterator >
  std::vector< double > qpp::abssq (InputIterator first, InputIterator last)

  *Computes the absolute values squared of an STL-like range of complex numbers.*
- template<typename Container >
  std::vector< double > qpp::abssq (const Container &c, typename std::enable_if< is_iterable< Container >::value >::type ∗=nullptr)

  *Computes the absolute values squared of an STL-like container.*
- template<typename Derived >
  std::vector< double > qpp::abssq (const Eigen::MatrixBase< Derived > &A)

  *Computes the absolute values squared of an Eigen expression.*
- template<typename InputIterator >
  std::iterator_traits< InputIterator >::value_type qpp::sum (InputIterator first, InputIterator last)

  *Element-wise sum of an STL-like range.*
- template<typename Container >
  Container::value_type qpp::sum (const Container &c, typename std::enable_if< is_iterable< Container >↩
  ::value >::type ∗=nullptr)

  *Element-wise sum of the elements of an STL-like container.*
- template<typename InputIterator >
  std::iterator_traits< InputIterator >::value_type qpp::prod (InputIterator first, InputIterator last)

  *Element-wise product of an STL-like range.*
- template<typename Container >
  Container::value_type qpp::prod (const Container &c, typename std::enable_if< is_iterable< Container >↩
  ::value >::type ∗=nullptr)

  *Element-wise product of the elements of an STL-like container.*
- template<typename Derived >
  dyn_col_vect< typename Derived::Scalar > qpp::rho2pure (const Eigen::MatrixBase< Derived > &A)

  *Finds the pure state representation of a matrix proportional to a projector onto a pure state.*
- std::vector< idx > qpp::complement (std::vector< idx > subsys, idx n)

  *Constructs the complement of a subsystem vector.*
- template<typename Derived >
  std::vector< double > qpp::rho2bloch (const Eigen::MatrixBase< Derived > &A)

  *Computes the 3-dimensional real Bloch vector corresponding to the qubit density matrix A.*
- cmat qpp::bloch2rho (const std::vector< double > &r)

  *Computes the density matrix corresponding to the 3-dimensional real Bloch vector r.*
- template<char... Bits>
  ket qpp::literals::operator"" _ket ()

  *Multi-partite qubit ket user-defined literal.*
- template<char... Bits>
  bra qpp::literals::operator"" _bra ()

  *Multi-partite qubit bra user-defined literal.*
- template<char... Bits>
  cmat qpp::literals::operator"" _prj ()

  *Multi-partite qubit projector user-defined literal.*
- template<class T >
  void qpp::internal::hash_combine (std::size_t &seed, const T &v)
- template<typename Derived >
  std::size_t qpp::hash_eigen (const Eigen::MatrixBase< Derived > &A, std::size_t seed=0)

  *Computes the hash of en Eigen matrix/vector/expression.*

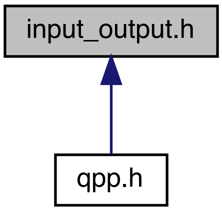
### 8.17.1 Detailed Description

Generic quantum computing functions.

## 8.18 input_output.h File Reference

Input/output functions.

This graph shows which files directly or indirectly include this file:




### Namespaces

- qpp

  *Quantum++ main namespace.*

### Functions

- template< typename Derived >
  internal::IOManipEigen qpp::disp (const Eigen::MatrixBase< Derived > &A, double chop=qpp::chop)

  *Eigen expression ostream manipulator.*
- internal::IOManipEigen qpp::disp (cplx z, double chop=qpp::chop)

  *Complex number ostream manipulator.*
- template< typename InputIterator >
  internal::IOManipRange< InputIterator > qpp::disp (InputIterator first, InputIterator last, const std::string &separator, const std::string &start="[", const std::string &end="]")

  *Range ostream manipulator.*
- template< typename Container >
  internal::IOManipRange< typename Container::const_iterator > qpp::disp (const Container &c, const std::string &separator, const std::string &start="[", const std::string &end="]", typename std::enable_if< is_iterable< Container >::value >::type *=nullptr)

  *Standard container ostream manipulator. The container must support std::begin(), std::end() and forward iteration.*
- template< typename PointerType >
  internal::IOManipPointer< PointerType > qpp::disp (const PointerType *p, idx N, const std::string &separator, const std::string &start="[", const std::string &end="]")

  *C-style pointer ostream manipulator.*
- template< typename Derived >
  void qpp::save (const Eigen::MatrixBase< Derived > &A, const std::string &fname)

  *Saves Eigen expression to a binary file (internal format) in double precision.*
- template< typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::load (const std::string &fname)

  *Loads Eigen matrix from a binary file (internal format) in double precision.*

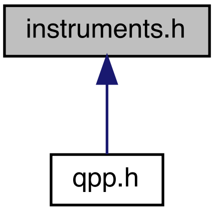
### 8.18.1 Detailed Description

Input/output functions.

## 8.19 instruments.h File Reference

Measurement functions.

This graph shows which files directly or indirectly include this file:




### Namespaces

- qpp

  *Quantum++ main namespace.*

### Functions

- template< typename Derived >
  dyn_col_vect< typename Derived::Scalar > qpp::ip (const Eigen::MatrixBase< Derived > &phi, const Eigen::MatrixBase< Derived > &psi, const std::vector< idx > &subsys, const std::vector< idx > &dims)

  *Generalized inner product.*
- template< typename Derived >
  dyn_col_vect< typename Derived::Scalar > qpp::ip (const Eigen::MatrixBase< Derived > &phi, const Eigen::MatrixBase< Derived > &psi, const std::vector< idx > &subsys, idx d=2)

  *Generalized inner product.*
- template< typename Derived >
  std::tuple< idx, std::vector< double >, std::vector< cmat > > qpp::measure (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat > &Ks)

  *Measures the state vector or density operator A using the set of Kraus operators Ks.*
- template< typename Derived >
  std::tuple< idx, std::vector< double >, std::vector< cmat > > qpp::measure (const Eigen::MatrixBase< Derived > &A, const std::initializer_list< cmat > &Ks)

  *Measures the state vector or density matrix A using the set of Kraus operators Ks.*
- template< typename Derived >
  std::tuple< idx, std::vector< double >, std::vector< cmat > > qpp::measure (const Eigen::MatrixBase< Derived > &A, const cmat &U)

*Measures the state vector or density matrix A in the orthonormal basis specified by the unitary matrix U.*

- template<typename Derived >
std::tuple< idx, std::vector< double >, std::vector< cmat > > qpp::measure (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat > &Ks, const std::vector< idx > &target, const std::vector< idx > &dims)

  *Measures the part subsys of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.*
- template<typename Derived >
std::tuple< idx, std::vector< double >, std::vector< cmat > > qpp::measure (const Eigen::MatrixBase< Derived > &A, const std::initializer_list< cmat > &Ks, const std::vector< idx > &target, const std::vector< idx > &dims)

  *Measures the part target of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.*
- template<typename Derived >
std::tuple< idx, std::vector< double >, std::vector< cmat > > qpp::measure (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat > &Ks, const std::vector< idx > &target, idx d=2)

  *Measures the part target of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.*
- template<typename Derived >
std::tuple< idx, std::vector< double >, std::vector< cmat > > qpp::measure (const Eigen::MatrixBase< Derived > &A, const std::initializer_list< cmat > &Ks, const std::vector< idx > &target, idx d=2)

  *Measures the part target of the multi-partite state vector or density matrix A using the set of Kraus operators Ks.*
- template<typename Derived >
std::tuple< idx, std::vector< double >, std::vector< cmat > > qpp::measure (const Eigen::MatrixBase< Derived > &A, const cmat &V, const std::vector< idx > &target, const std::vector< idx > &dims)

  *Measures the part target of the multi-partite state vector or density matrix A in the orthonormal basis or rank-1 projectors specified by the columns of the matrix V.*
- template<typename Derived >
std::tuple< idx, std::vector< double >, std::vector< cmat > > qpp::measure (const Eigen::MatrixBase< Derived > &A, const cmat &V, const std::vector< idx > &target, idx d=2)

  *Measures the part target of the multi-partite state vector or density matrix A in the orthonormal basis or rank-1 projectors specified by the columns of the matrix V.*
- template<typename Derived >
std::tuple< std::vector< idx >, double, cmat > qpp::measure_seq (const Eigen::MatrixBase< Derived > &A, std::vector< idx > target, std::vector< idx > dims)

  *Sequentially measures the part target of the multi-partite state vector or density matrix A in the computational basis.*
- template<typename Derived >
std::tuple< std::vector< idx >, double, cmat > qpp::measure_seq (const Eigen::MatrixBase< Derived > &A, std::vector< idx > target, idx d=2)

  *Sequentially measures the part target of the multi-partite state vector or density matrix A in the computational basis.*

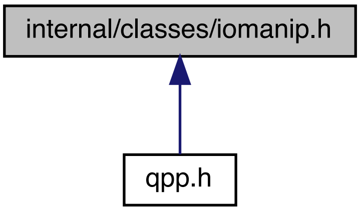
### 8.19.1 Detailed Description

Measurement functions.

## 8.20 internal/classes/iomanip.h File Reference

Input/output manipulators.

This graph shows which files directly or indirectly include this file:




## Classes

- class qpp::internal::IOManipRange< InputIterator >
- class qpp::internal::IOManipPointer< PointerType >
- class qpp::internal::IOManipEigen

## Namespaces

- qpp

    *Quantum++ main namespace.*
- qpp::internal

    *Internal utility functions, do not use them directly or modify them.*

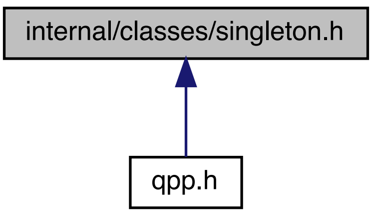
### 8.20.1 Detailed Description

Input/output manipulators.

## 8.21 internal/classes/singleton.h File Reference

Singleton pattern via CRTP.

This graph shows which files directly or indirectly include this file:



## Classes

- class qpp::internal::Singleton< T >

  *Singleton policy class, used internally to implement the singleton pattern via CRTP (Curiously recurring template pattern)*

## Namespaces

- qpp

  *Quantum++ main namespace.*
- qpp::internal

  *Internal utility functions, do not use them directly or modify them.*

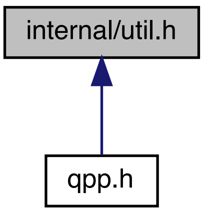
### 8.21.1 Detailed Description

Singleton pattern via CRTP.

# 8.22 internal/util.h File Reference

Internal utility functions.

This graph shows which files directly or indirectly include this file:




## Classes

- struct qpp::internal::Display_Impl_

## Namespaces

- qpp

  *Quantum++ main namespace.*
- qpp::internal

  *Internal utility functions, do not use them directly or modify them.*

### Functions

- void qpp::internal::n2multiidx (idx n, idx numdims, const idx ∗const dims, idx ∗result) noexcept
- idx qpp::internal::multiidx2n (const idx ∗const midx, idx numdims, const idx ∗const dims) noexcept
- template< typename Derived >
  bool qpp::internal::check_square_mat (const Eigen::MatrixBase< Derived > &A)
- template< typename Derived >
  bool qpp::internal::check_vector (const Eigen::MatrixBase< Derived > &A)
- template< typename Derived >
  bool qpp::internal::check_rvector (const Eigen::MatrixBase< Derived > &A)
- template< typename Derived >
  bool qpp::internal::check_cvector (const Eigen::MatrixBase< Derived > &A)
- template< typename T >
  bool qpp::internal::check_nonzero_size (const T &x) noexcept
- template< typename T1 , typename T2 >
  bool qpp::internal::check_matching_sizes (const T1 &lhs, const T2 &rhs) noexcept
- bool qpp::internal::check_dims (const std::vector< idx > &dims)
- template< typename Derived >
  bool qpp::internal::check_dims_match_mat (const std::vector< idx > &dims, const Eigen::MatrixBase< Derived > &A)
- template< typename Derived >
  bool qpp::internal::check_dims_match_cvect (const std::vector< idx > &dims, const Eigen::MatrixBase< Derived > &A)
- template< typename Derived >
  bool qpp::internal::check_dims_match_rvect (const std::vector< idx > &dims, const Eigen::MatrixBase< Derived > &A)
- bool qpp::internal::check_eq_dims (const std::vector< idx > &dims, idx dim) noexcept
- bool qpp::internal::check_no_duplicates (std::vector< idx > v)
- bool qpp::internal::check_subsys_match_dims (const std::vector< idx > &subsys, const std::vector< idx > &dims)
- template< typename Derived >
  bool qpp::internal::check_qubit_matrix (const Eigen::MatrixBase< Derived > &A) noexcept
- template< typename Derived >
  bool qpp::internal::check_qubit_cvector (const Eigen::MatrixBase< Derived > &A) noexcept
- template< typename Derived >
  bool qpp::internal::check_qubit_rvector (const Eigen::MatrixBase< Derived > &A) noexcept
- template< typename Derived >
  bool qpp::internal::check_qubit_vector (const Eigen::MatrixBase< Derived > &A) noexcept
- bool qpp::internal::check_perm (const std::vector< idx > &perm)
- template< typename Derived1 , typename Derived2 >
  dyn_mat< typename Derived1::Scalar > qpp::internal::kron2 (const Eigen::MatrixBase< Derived1 > &A, const Eigen::MatrixBase< Derived2 > &B)
- template< typename Derived1 , typename Derived2 >
  dyn_mat< typename Derived1::Scalar > qpp::internal::dirsum2 (const Eigen::MatrixBase< Derived1 > &A, const Eigen::MatrixBase< Derived2 > &B)
- template< typename T >
  void qpp::internal::variadic_vector_emplace (std::vector< T > &)
- template< typename T , typename First , typename... Args>
  void qpp::internal::variadic_vector_emplace (std::vector< T > &v, First &&first, Args &&... args)
- idx qpp::internal::get_num_subsys (idx D, idx d)
- idx qpp::internal::get_dim_subsys (idx sz, idx N)

## 8.22.1 Detailed Description

Internal utility functions.

## 8.23 MATLAB/matlab.h File Reference

Input/output interfacing with MATLAB.

```
#include "mat.h"
#include "mex.h"
```

### Namespaces

- qpp

  *Quantum++ main namespace.*

### Functions

- template<typename Derived >
  std::enable_if< std::is_same< typename Derived::Scalar, cplx >::value, dyn_mat< cplx > >::type qpp::loadMATLAB (const std::string &mat_file, const std::string &var_name)

  *Loads a complex Eigen dynamic matrix from a MATLAB .mat file,.*
- template<typename Derived >
  std::enable_if<!std::is_same< typename Derived::Scalar, cplx >::value, dyn_mat< typename Derived::Scalar > >::type qpp::loadMATLAB (const std::string &mat_file, const std::string &var_name)

  *Loads a non-complex Eigen dynamic matrix from a MATLAB .mat file,.*
- template<typename Derived >
  std::enable_if< std::is_same< typename Derived::Scalar, cplx >::value >::type qpp::saveMATLAB (const Eigen::MatrixBase< Derived > &A, const std::string &mat_file, const std::string &var_name, const std::string &mode)

  *Saves a complex Eigen dynamic matrix to a MATLAB .mat file,.*
- template<typename Derived >
  std::enable_if< !std::is_same< typename Derived::Scalar, cplx >::value >::type qpp::saveMATLAB (const Eigen::MatrixBase< Derived > &A, const std::string &mat_file, const std::string &var_name, const std::string &mode)

  *Saves a non-complex Eigen dynamic matrix to a MATLAB .mat file,.*

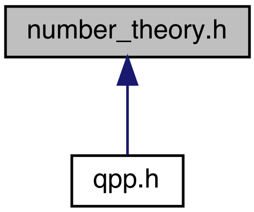
### 8.23.1 Detailed Description

Input/output interfacing with MATLAB.

## 8.24 number_theory.h File Reference

Number theory functions.

This graph shows which files directly or indirectly include this file:




## Namespaces

- qpp

  *Quantum++ main namespace.*

## Functions

- std::vector< int > qpp::x2contfrac (double x, idx N, idx cut=1e5)

  *Simple continued fraction expansion.*
- double qpp::contfrac2x (const std::vector< int > &cf, idx N=idx(-1))

  *Real representation of a simple continued fraction.*
- bigint qpp::gcd (bigint a, bigint b)

  *Greatest common divisor of two integers.*
- bigint qpp::gcd (const std::vector< bigint > &as)

  *Greatest common divisor of a list of integers.*
- bigint qpp::lcm (bigint a, bigint b)

  *Least common multiple of two integers.*
- bigint qpp::lcm (const std::vector< bigint > &as)

  *Least common multiple of a list of integers.*
- std::vector< idx > qpp::invperm (const std::vector< idx > &perm)

  *Inverse permutation.*
- std::vector< idx > qpp::compperm (const std::vector< idx > &perm, const std::vector< idx > &sigma)

  *Compose permutations.*
- std::vector< bigint > qpp::factors (bigint a)

  *Prime factor decomposition.*
- bigint qpp::modmul (bigint a, bigint b, bigint p)

  *Modular multiplication without overflow.*
- bigint qpp::modpow (bigint a, bigint n, bigint p)

  *Fast integer power modulo p based on the SQUARE-AND-MULTIPLY algorithm.*
- std::tuple< bigint, bigint, bigint > qpp::egcd (bigint a, bigint b)

  *Extended greatest common divisor of two integers.*
- bigint qpp::modinv (bigint a, bigint p)

  *Modular inverse of a mod p.*
- bool qpp::isprime (bigint p, idx k=80)

*Primality test based on the Miller-Rabin's algorithm.*
- bigint qpp::randprime (bigint a, bigint b, idx N=1000)

  *Generates a random big prime uniformly distributed in the interval [a, b].*
- std::vector< std::pair< int, int > > qpp::convergents (const std::vector< int > &cf)

  *Convergents.*
- std::vector< std::pair< int, int > > qpp::convergents (double x, idx N)

  *Convergents.*

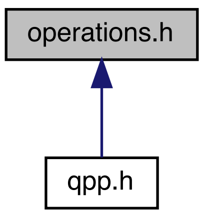
### 8.24.1 Detailed Description

Number theory functions.

## 8.25 operations.h File Reference

Quantum operation functions.

This graph shows which files directly or indirectly include this file:




### Namespaces

- qpp

  *Quantum++ main namespace.*

### Functions

- template<typename Derived1 , typename Derived2 >
  dyn_mat< typename Derived1::Scalar > qpp::applyCTRL (const Eigen::MatrixBase< Derived1 > &state, const Eigen::MatrixBase< Derived2 > &A, const std::vector< idx > &ctrl, const std::vector< idx > &target, const std::vector< idx > &dims)

  *Applies the controlled-gate A to the part target of the multi-partite state vector or density matrix state.*
- template<typename Derived1 , typename Derived2 >
  dyn_mat< typename Derived1::Scalar > qpp::applyCTRL (const Eigen::MatrixBase< Derived1 > &state, const Eigen::MatrixBase< Derived2 > &A, const std::vector< idx > &ctrl, const std::vector< idx > &target, idx d=2)

*Applies the controlled-gate A to the part target of the multi-partite state vector or density matrix state.*

- template< typename Derived1 , typename Derived2 >
  dyn_mat< typename Derived1::Scalar > qpp::apply (const Eigen::MatrixBase< Derived1 > &state, const Eigen::MatrixBase< Derived2 > &A, const std::vector< idx > &target, const std::vector< idx > &dims)

  *Applies the gate A to the part target of the multi-partite state vector or density matrix state.*
- template< typename Derived1 , typename Derived2 >
  dyn_mat< typename Derived1::Scalar > qpp::apply (const Eigen::MatrixBase< Derived1 > &state, const Eigen::MatrixBase< Derived2 > &A, const std::vector< idx > &target, idx d=2)

  *Applies the gate A to the part target of the multi-partite state vector or density matrix state.*
- template< typename Derived >
  cmat qpp::apply (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat > &Ks)

  *Applies the channel specified by the set of Kraus operators Ks to the density matrix A.*
- template< typename Derived >
  cmat qpp::apply (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat > &Ks, const std::vector< idx > &target, const std::vector< idx > &dims)

  *Applies the channel specified by the set of Kraus operators Ks to the part target of the multi-partite density matrix A.*
- template< typename Derived >
  cmat qpp::apply (const Eigen::MatrixBase< Derived > &A, const std::vector< cmat > &Ks, const std::vector< idx > &target, idx d=2)

  *Applies the channel specified by the set of Kraus operators Ks to the part target of the multi-partite density matrix A.*
- cmat qpp::kraus2super (const std::vector< cmat > &Ks)

  *Superoperator matrix.*
- cmat qpp::kraus2choi (const std::vector< cmat > &Ks)

  *Choi matrix.*
- std::vector< cmat > qpp::choi2kraus (const cmat &A)

  *Orthogonal Kraus operators from Choi matrix.*
- cmat qpp::choi2super (const cmat &A)

  *Converts Choi matrix to superoperator matrix.*
- cmat qpp::super2choi (const cmat &A)

  *Converts superoperator matrix to Choi matrix.*
- template< typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::ptrace1 (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)

  *Partial trace.*
- template< typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::ptrace1 (const Eigen::MatrixBase< Derived > &A, idx d=2)

  *Partial trace.*
- template< typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::ptrace2 (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &dims)

  *Partial trace.*
- template< typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::ptrace2 (const Eigen::MatrixBase< Derived > &A, idx d=2)

  *Partial trace.*
- template< typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::ptrace (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &target, const std::vector< idx > &dims)

  *Partial trace.*
- template< typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::ptrace (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &target, idx d=2)

  *Partial trace.*

- template< typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::ptranspose (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &target, const std::vector< idx > &dims)

  *Partial transpose.*
- template< typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::ptranspose (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &target, idx d=2)

  *Partial transpose.*
- template< typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::syspermute (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &perm, const std::vector< idx > &dims)

  *Subsystem permutation.*
- template< typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::syspermute (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &perm, idx d=2)

  *Subsystem permutation.*
- template< typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::applyQFT (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &target, idx d=2, bool swap=true)

  *Applies the qudit quantum Fourier transform to the part target of the multi-partite state vector or density matrix A.*
- template< typename Derived >
  dyn_mat< typename Derived::Scalar > qpp::applyTFQ (const Eigen::MatrixBase< Derived > &A, const std::vector< idx > &target, idx d=2, bool swap=true)

  *Applies the inverse (adjoint) qudit quantum Fourier transform to the part target of the multi-partite state vector or density matrix A.*
- template< typename Derived >
  dyn_col_vect< typename Derived::Scalar > qpp::TFQ (const Eigen::MatrixBase< Derived > &A, idx d=2, bool swap=true)

  *Inverse (adjoint) qudit quantum Fourier transform.*
- template< typename Derived >
  dyn_col_vect< typename Derived::Scalar > qpp::QFT (const Eigen::MatrixBase< Derived > &A, idx d=2, bool swap=true)

  *Qudit quantum Fourier transform.*

### 8.25.1 Detailed Description

Quantum operation functions.

## 8.26 qpp.h File Reference

Quantum++ main header file, includes all other necessary headers.

```
#include <algorithm>
#include <cassert>
#include <chrono>
#include <cmath>
#include <complex>
#include <cstdlib>
#include <cstring>
#include <exception>
#include <fstream>
```

```
#include <functional>
#include <initializer_list>
#include <iomanip>
#include <iterator>
#include <limits>
#include <memory>
#include <numeric>
#include <ostream>
#include <random>
#include <sstream>
#include <stdexcept>
#include <string>
#include <tuple>
#include <type_traits>
#include <unordered_map>
#include <utility>
#include <vector>
#include <Eigen/Dense>
#include <Eigen/SVD>
#include "types.h"
#include "classes/exception.h"
#include "constants.h"
#include "traits.h"
#include "classes/idisplay.h"
#include "internal/util.h"
#include "internal/classes/iomanip.h"
#include "input_output.h"
#include "internal/classes/singleton.h"
#include "classes/random_devices.h"
#include "random.h"
#include "number_theory.h"
#include "functions.h"
#include "classes/init.h"
#include "classes/codes.h"
#include "classes/gates.h"
#include "classes/states.h"
#include "statistics.h"
#include "operations.h"
#include "entropies.h"
#include "entanglement.h"
#include "classes/timer.h"
#include "instruments.h"
#include "classes/reversible.h"
#include "classes/noise.h"
#include "classes/circuits/circuits.h"
#include "classes/circuits/engines.h"
```

## Namespaces

- qpp

  *Quantum++ main namespace.*

## Macros

- #define QPP_UNUSED_

### 8.26.1 Detailed Description

Quantum++ main header file, includes all other necessary headers.

### 8.26.2 Macro Definition Documentation

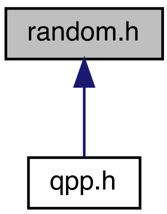
#### 8.26.2.1 QPP_UNUSED_

```
#define QPP_UNUSED_
```

## 8.27 random.h File Reference

Randomness-related functions.

This graph shows which files directly or indirectly include this file:




### Namespaces

- qpp

  *Quantum++ main namespace.*

## Functions

- double qpp::rand (double a, double b)

  *Generates a random real number uniformly distributed in the interval [a, b)*
- bigint qpp::rand (bigint a, bigint b)

  *Generates a random big integer uniformly distributed in the interval [a, b].*
- idx qpp::randidx (idx a=std::numeric_limits< idx >::min(), idx b=std::numeric_limits< idx >::max())

  *Generates a random index (idx) uniformly distributed in the interval [a, b].*
- template<typename Derived >
  Derived qpp::rand (idx rows QPP_UNUSED_, idx cols QPP_UNUSED_, double a QPP_UNUSED_=0, double b QPP_UNUSED_=1)

  *Generates a random matrix with entries uniformly distributed in the interval [a, b)*
- template<>
  dmat qpp::rand (idx rows, idx cols, double a, double b)

  *Generates a random real matrix with entries uniformly distributed in the interval [a, b), specialization for double matrices (qpp::dmat)*
- template<>
  cmat qpp::rand (idx rows, idx cols, double a, double b)

  *Generates a random complex matrix with entries (both real and imaginary) uniformly distributed in the interval [a, b), specialization for complex matrices (qpp::cmat)*
- template<typename Derived >
  Derived qpp::randn (idx rows QPP_UNUSED_, idx cols QPP_UNUSED_, double mean QPP_UNUSED_=0, double sigma QPP_UNUSED_=1)

  *Generates a random matrix with entries normally distributed in N(mean, sigma)*
- template<>
  dmat qpp::randn (idx rows, idx cols, double mean, double sigma)

  *Generates a random real matrix with entries normally distributed in N(mean, sigma), specialization for double matrices (qpp::dmat)*
- template<>
  cmat qpp::randn (idx rows, idx cols, double mean, double sigma)

  *Generates a random complex matrix with entries (both real and imaginary) normally distributed in N(mean, sigma), specialization for complex matrices (qpp::cmat)*
- double qpp::randn (double mean=0, double sigma=1)

  *Generates a random real number (double) normally distributed in N(mean, sigma)*
- cmat qpp::randU (idx D=2)

  *Generates a random unitary matrix.*
- cmat qpp::randV (idx Din, idx Dout)

  *Generates a random isometry matrix.*
- std::vector< cmat > qpp::randkraus (idx N, idx D=2)

  *Generates a set of random Kraus operators.*
- cmat qpp::randH (idx D=2)

  *Generates a random Hermitian matrix.*
- ket qpp::randket (idx D=2)

  *Generates a random normalized ket (pure state vector)*
- cmat qpp::randrho (idx D=2)

  *Generates a random density matrix.*
- std::vector< idx > qpp::randperm (idx N)

  *Generates a random uniformly distributed permutation.*
- std::vector< double > qpp::randprob (idx N)

  *Generates a random probability vector uniformly distributed over the probability simplex.*

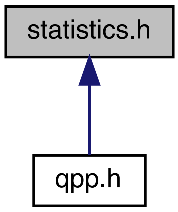
### 8.27.1 Detailed Description

Randomness-related functions.

## 8.28 statistics.h File Reference

Statistics functions.

This graph shows which files directly or indirectly include this file:




### Namespaces

- qpp
  *Quantum++ main namespace.*

### Functions

- std::vector< double > qpp::uniform (idx N)
  *Uniform probability distribution vector.*
- std::vector< double > qpp::marginalX (const dmat &probXY)
  *Marginal distribution.*
- std::vector< double > qpp::marginalY (const dmat &probXY)
  *Marginal distribution.*
- template<typename Container >
  double qpp::avg (const std::vector< double > &prob, const Container &X, typename std::enable_if< is_iterable< Container >::value >::type ∗=nullptr)
  *Average.*
- template<typename Container >
  double qpp::cov (const dmat &probXY, const Container &X, const Container &Y, typename std::enable_if< is_iterable< Container >::value >::type ∗=nullptr)
  *Covariance.*
- template<typename Container >
  double qpp::var (const std::vector< double > &prob, const Container &X, typename std::enable_if< is_iterable< Container >::value >::type ∗=nullptr)
  *Variance.*

- template<typename Container >
double qpp::sigma (const std::vector< double > &prob, const Container &X, typename std::enable_if< is_↩
iterable< Container >::value >::type ∗=nullptr)

    *Standard deviation.*
- template<typename Container >
double qpp::cor (const dmat &probXY, const Container &X, const Container &Y, typename std::enable_if<
is_iterable< Container >::value >::type ∗=nullptr)

    *Correlation.*

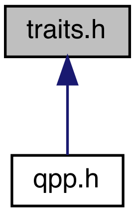
### 8.28.1 Detailed Description

Statistics functions.

## 8.29 traits.h File Reference

Type traits.

This graph shows which files directly or indirectly include this file:




### Classes

- struct qpp::make_void< Ts >

    *Helper for qpp::to_void<> alias template.*
- struct qpp::is_iterable< T, typename >

    *Checks whether T is compatible with an STL-like iterable container.*
- struct qpp::is_iterable< T, to_void< decltype(std::declval< T >().begin()), decltype(std::declval< T >().end()), decltype(∗(std::

    *Checks whether T is compatible with an STL-like iterable container, specialization for STL-like iterable containers.*
- struct qpp::is_matrix_expression< Derived >

    *Checks whether the type is an Eigen matrix expression.*
- struct qpp::is_complex< T >

    *Checks whether the type is a complex type.*
- struct qpp::is_complex< std::complex< T > >

    *Checks whether the type is a complex number type, specialization for complex types.*

### Namespaces

- qpp

  *Quantum++ main namespace.*

### Typedefs

- template< typename... Ts>
  using qpp::to_void = typename make_void< Ts... >::type

  *Alias template that implements the proposal for void_t.*

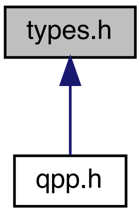
### 8.29.1 Detailed Description

Type traits.

## 8.30 types.h File Reference

Type aliases.

This graph shows which files directly or indirectly include this file:




### Namespaces

- qpp

  *Quantum++ main namespace.*

## Typedefs

- using qpp::idx = std::size_t

    *Non-negative integer index, make sure you use an unsigned type.*
- using qpp::bigint = long long int

    *Big integer.*
- using qpp::cplx = std::complex< double >

    *Complex number in double precision.*
- using qpp::ket = Eigen::VectorXcd

    *Complex (double precision) dynamic Eigen column vector.*
- using qpp::bra = Eigen::RowVectorXcd

    *Complex (double precision) dynamic Eigen row vector.*
- using qpp::cmat = Eigen::MatrixXcd

    *Complex (double precision) dynamic Eigen matrix.*
- using qpp::dmat = Eigen::MatrixXd

    *Real (double precision) dynamic Eigen matrix.*
- template< typename Scalar >
  using qpp::dyn_mat = Eigen::Matrix< Scalar, Eigen::Dynamic, Eigen::Dynamic >

    *Dynamic Eigen matrix over the field specified by Scalar.*
- template< typename Scalar >
  using qpp::dyn_col_vect = Eigen::Matrix< Scalar, Eigen::Dynamic, 1 >

    *Dynamic Eigen column vector over the field specified by Scalar.*
- template< typename Scalar >
  using qpp::dyn_row_vect = Eigen::Matrix< Scalar, 1, Eigen::Dynamic >

    *Dynamic Eigen row vector over the field specified by Scalar.*

### 8.30.1 Detailed Description

Type aliases.

# Index

~Bit_circuit
qpp::Bit_circuit, 132
~Codes
qpp::Codes, 139
~Dynamic_bitset
qpp::Dynamic_bitset, 161
~Gates
qpp::Gates, 176
~IDisplay
qpp::IDisplay, 191
~IJSON
qpp::IJSON, 193
~Init
qpp::Init, 196
~NoiseBase
qpp::NoiseBase, 242
~QCircuit
qpp::QCircuit, 275
~QEngine
qpp::QEngine, 301
~RandomDevices
qpp::RandomDevices, 327
~Singleton
qpp::internal::Singleton, 331
~States
qpp::States, 337
~Timer
qpp::Timer, 348

A_
qpp::internal::IOManipEigen, 200
absm
qpp, 29
abssq
qpp, 29, 30
add_hash_
qpp::QCircuit, 275
adjoint
qpp, 30
all
qpp::Dynamic_bitset, 161
anticomm
qpp, 31
any
qpp::Dynamic_bitset, 161
apply
qpp, 31–33
applyCTRL
qpp, 34
applyQFT
qpp, 35
applyTFQ
qpp, 36
avg
qpp, 36

b00
qpp::States, 339
b01
qpp::States, 340
b10
qpp::States, 340
b11
qpp::States, 340
bCNOT_
qpp::Bit_circuit, 136
bFRED_
qpp::Bit_circuit, 136
bNOT_
qpp::Bit_circuit, 136
bSWAP_
qpp::Bit_circuit, 136
bTOF_
qpp::Bit_circuit, 136
begin
qpp::QCircuit, 275
bigint
qpp, 26
Bit_circuit
qpp::Bit_circuit, 131
bloch2rho
qpp, 36
bra
qpp, 26
btotal_
qpp::Bit_circuit, 136

c_reg_
qpp::QCircuit::MeasureStep, 237
cCTRL_custom
qpp::QCircuit, 278
cCTRL
qpp::QCircuit, 276, 277
CNOTba
qpp::Gates, 183
CNOT
qpp::Bit_circuit, 132
qpp::Gates, 183
CTRL_custom
qpp::QCircuit, 281

CTRL
qpp::Gates, 176
qpp::QCircuit, 279, 280
cbegin
qpp::QCircuit, 276
cend
qpp::QCircuit, 278
check_cvector
qpp::internal, 120
check_dims
qpp::internal, 120
check_dims_match_cvect
qpp::internal, 120
check_dims_match_mat
qpp::internal, 120
check_dims_match_rvect
qpp::internal, 120
check_eq_dims
qpp::internal, 121
check_matching_sizes
qpp::internal, 121
check_no_duplicates
qpp::internal, 121
check_nonzero_size
qpp::internal, 121
check_perm
qpp::internal, 121
check_qubit_cvector
qpp::internal, 121
check_qubit_matrix
qpp::internal, 122
check_qubit_rvector
qpp::internal, 122
check_qubit_vector
qpp::internal, 122
check_rvector
qpp::internal, 122
check_square_mat
qpp::internal, 122
check_subsys_match_dims
qpp::internal, 122
check_vector
qpp::internal, 123
choi2kraus
qpp, 37
choi2super
qpp, 37
chop
qpp, 116
chop_
qpp::internal::IOManipEigen, 201
classes/circuits/circuits.h, 363
classes/circuits/engines.h, 364
classes/codes.h, 365
classes/exception.h, 365
classes/gates.h, 367
classes/idisplay.h, 368
classes/init.h, 369
classes/noise.h, 370
classes/random_devices.h, 371
classes/reversible.h, 371
classes/states.h, 372
classes/timer.h, 373
cmat
qpp, 27
cmat_hash_tbl_
qpp::QCircuit, 295
Codes
qpp::Codes, 139
codeword
qpp::Codes, 139
comm
qpp, 38
complement
qpp, 38
compperm
qpp, 39
compute_probs_
qpp::NoiseBase, 243
compute_state_
qpp::NoiseBase, 243
concurrence
qpp, 39
conjugate
qpp, 40
const_iterator
qpp::QCircuit, 273
constants.h, 374
contfrac2x
qpp, 40
convergents
qpp, 40, 41
cor
qpp, 41
cosm
qpp, 42
count
qpp::Dynamic_bitset, 161
count_
qpp::Bit_circuit, 136
qpp::QCircuit, 296
cov
qpp, 42
cplx
qpp, 27
ctrl_
qpp::QCircuit::GateStep, 187
CustomException
qpp::exception::CustomException, 141
cwise
qpp, 43
CZ
qpp::Gates, 184

d_
qpp::NoiseBase, 246
qpp::QCircuit, 296

data
qpp::Dynamic_bitset, 161
depth_
qpp::Bit_circuit, 137
det
qpp, 43
difference_type
qpp::QCircuit::iterator, 214
dirsum
qpp, 44, 45
dirsum2
qpp::internal, 123
dirsumpow
qpp, 45
disp
qpp, 46–48
display
qpp::Dynamic_bitset, 162
qpp::IDisplay, 191
qpp::QCircuit, 281
qpp::QCircuit::iterator::value_type_, 359
qpp::QEngine, 301
qpp::Timer, 348
qpp::internal::IOManipEigen, 200
qpp::internal::IOManipPointer, 203
qpp::internal::IOManipRange, 206
display_impl_
qpp::internal::Display_Impl_, 155
dits_
qpp::QEngine, 307
dmat
qpp, 27
dyn_col_vect
qpp, 27
dyn_mat
qpp, 27
dyn_row_vect
qpp, 28
Dynamic_bitset
qpp::Dynamic_bitset, 160

ee
qpp, 116
egcd
qpp, 48
eig
qpp, 49
elem_
qpp::QCircuit::iterator, 218
end
qpp::QCircuit, 281, 282
end_
qpp::Timer, 350
qpp::internal::IOManipPointer, 203
qpp::internal::IOManipRange, 207
entanglement
qpp, 49, 50
entanglement.h, 375
entropies.h, 376
entropy
qpp, 50, 51
evals
qpp, 51
evects
qpp, 51
Exception
qpp::exception::DimsInvalid, 144
qpp::exception::DimsMismatchCvector, 146
qpp::exception::DimsMismatchMatrix, 148
qpp::exception::DimsMismatchRvector, 150
qpp::exception::DimsMismatchVector, 152
qpp::exception::DimsNotEqual, 154
qpp::exception::Duplicates, 157
qpp::exception::Exception, 172
qpp::exception::InvalidIterator, 198
qpp::exception::MatrixMismatchSubsys, 220
qpp::exception::MatrixNotCvector, 222
qpp::exception::MatrixNotRvector, 224
qpp::exception::MatrixNotSquare, 226
qpp::exception::MatrixNotSquareNorCvector, 228
qpp::exception::MatrixNotSquareNorRvector, 230
qpp::exception::MatrixNotSquareNorVector, 232
qpp::exception::MatrixNotVector, 234
qpp::exception::NoCodeword, 239
qpp::exception::NotBipartite, 249
qpp::exception::NotImplemented, 251
qpp::exception::NotQubitCvector, 253
qpp::exception::NotQubitMatrix, 255
qpp::exception::NotQubitRvector, 257
qpp::exception::NotQubitSubsys, 259
qpp::exception::NotQubitVector, 261
qpp::exception::OutOfRange, 263
qpp::exception::PermInvalid, 265
qpp::exception::PermMismatchDims, 267
qpp::exception::QuditAlreadyMeasured, 322
qpp::exception::SizeMismatch, 333
qpp::exception::SubsysMismatchDims, 345
qpp::exception::TypeMismatch, 352
qpp::exception::UndefinedType, 354
qpp::exception::Unknown, 356
qpp::exception::ZeroSize, 362
execute
qpp::QEngine, 301, 302
qpp::QNoisyEngine, 310, 311
expandout
qpp::Gates, 177, 178
experimental/experimental.h, 378
expm
qpp, 52

FRED
qpp::Bit_circuit, 132
qpp::Gates, 184
factors
qpp, 52
Fd
qpp::Gates, 179
fill_Ks_

qpp::QuditDepolarizingNoise, 325
fill_probs_
qpp::QuditDepolarizingNoise, 325
first_
qpp::internal::IOManipRange, 207
flip
qpp::Dynamic_bitset, 162
functions.h, 378
funm
qpp, 53

GHZ
qpp::States, 340
gate
qpp::QCircuit, 282, 283
gate_custom
qpp::QCircuit, 283
gate_fan
qpp::QCircuit, 284, 285
gate_hash_
qpp::QCircuit::GateStep, 187
gate_type_
qpp::QCircuit::GateStep, 188
GateStep
qpp::QCircuit::GateStep, 187
GateType
qpp::QCircuit, 273
Gates
qpp::Gates, 176
gates_
qpp::QCircuit, 296
gates_ip_
qpp::QCircuit::iterator::value_type_, 359
gcd
qpp, 53
gconcurrence
qpp, 54
generated_
qpp::NoiseBase, 246
get
qpp::Dynamic_bitset, 163
get_Ks
qpp::NoiseBase, 244
get_circuit
qpp::QEngine, 302
get_cmat_hash_tbl_
qpp::QCircuit, 285
get_d
qpp::NoiseBase, 243
qpp::QCircuit, 285
get_dim_subsys
qpp::internal, 123
get_dit
qpp::QEngine, 302
get_dits
qpp::QEngine, 303
get_duration
qpp::Timer, 349
get_gate_count
qpp::Bit_circuit, 133
qpp::QCircuit, 285
get_gate_depth
qpp::Bit_circuit, 133
qpp::QCircuit, 286
get_gates_
qpp::QCircuit, 286
get_instance
qpp::internal::Singleton, 331
get_last_idx
qpp::NoiseBase, 244
get_last_K
qpp::NoiseBase, 244
get_last_p
qpp::NoiseBase, 244
get_measured
qpp::QCircuit, 286, 287
qpp::QEngine, 303
get_measurement_count
qpp::QCircuit, 287
get_measurements_
qpp::QCircuit, 288
get_name
qpp::Gates, 179
qpp::QCircuit, 288
get_nc
qpp::QCircuit, 288
get_noise_results
qpp::QNoisyEngine, 311
get_non_measured
qpp::QCircuit, 288
qpp::QEngine, 303
get_nop_count
qpp::QCircuit, 289
get_nq
qpp::QCircuit, 289
get_num_subsys
qpp::internal, 123
get_prng
qpp::RandomDevices, 328
get_probs
qpp::NoiseBase, 245
qpp::QEngine, 304
get_psi
qpp::QEngine, 304
get_relative_pos_
qpp::QEngine, 304
get_step_count
qpp::QCircuit, 289
get_thread_local_instance
qpp::internal::Singleton, 331
grams
qpp, 54, 55

H
qpp::Gates, 184
hash_combine
qpp::internal, 123
hash_eigen

qpp, 56
heig
qpp, 56
hevals
qpp, 57
hevects
qpp, 57

i_
qpp::NoiseBase, 247
IDisplay
qpp::IDisplay, 190, 191
IJSON
qpp::IJSON, 193
IOManipEigen
qpp::internal::IOManipEigen, 200
IOManipPointer
qpp::internal::IOManipPointer, 202, 203
IOManipRange
qpp::internal::IOManipRange, 206
Id
qpp::Gates, 179
Id2
qpp::Gates, 184
idx
qpp, 28
index_
qpp::Dynamic_bitset, 163
infty
qpp, 116
Init
qpp::Init, 196
input_output.h, 383
instruments.h, 384
internal/classes/iomanip.h, 385
internal/classes/singleton.h, 386
internal/util.h, 387
internal::Singleton< const Codes >
qpp::Codes, 139
internal::Singleton< const Gates >
qpp::Gates, 183
internal::Singleton< const Init >
qpp::Init, 196
internal::Singleton< const States >
qpp::States, 339
internal::Singleton< RandomDevices >
qpp::RandomDevices, 329
inverse
qpp, 57
invperm
qpp, 58
ip
qpp, 58, 59
ip_
qpp::QCircuit::iterator::value_type_, 360
isprime
qpp, 59
iterator
qpp::QCircuit::iterator, 215
iterator_category
qpp::QCircuit::iterator, 214

jn
qpp::States, 337

ket
qpp, 28
kraus2choi
qpp, 59
kraus2super
qpp, 60
kron
qpp, 60–62
kron2
qpp::internal, 123
kronpow
qpp, 62
Ks_
qpp::NoiseBase, 247

last_
qpp::internal::IOManipRange, 207
lcm
qpp, 63
load
qpp, 64
qpp::RandomDevices, 328
loadMATLAB
qpp, 64, 65
logdet
qpp, 66
logm
qpp, 66
lognegativity
qpp, 67

MATLAB/matlab.h, 389
MODMUL
qpp::Gates, 180
marginalX
qpp, 67
marginalY
qpp, 69
mats_hash_
qpp::QCircuit::MeasureStep, 237
maxn
qpp, 116
measure
qpp, 69–74
measure_seq
qpp, 74, 75
MeasureStep
qpp::QCircuit::MeasureStep, 236
MeasureType
qpp::QCircuit, 274
measured_
qpp::QCircuit, 296
measurement_count_

qpp::QCircuit, 296
measurement_type_
qpp::QCircuit::MeasureStep, 237
measurements_
qpp::QCircuit, 296
measurements_ip_
qpp::QCircuit::iterator::value_type_, 360
measureV
qpp::QCircuit, 289, 290
measureZ
qpp::QCircuit, 290
mes
qpp::States, 337
minus
qpp::States, 338
mket
qpp, 75, 76
modinv
qpp, 76
modmul
qpp, 77
modpow
qpp, 77
mprj
qpp, 78
msg_
qpp::exception::Exception, 173
multiidx2n
qpp, 79
qpp::internal, 124

n2multiidx
qpp, 79
qpp::internal, 124
N_
qpp::internal::IOManipPointer, 204
n_
qpp::Dynamic_bitset, 168
NOT
qpp::Bit_circuit, 134
name_
qpp::QCircuit, 297
qpp::QCircuit::GateStep, 188
qpp::QCircuit::MeasureStep, 237
nc_
qpp::QCircuit, 297
negativity
qpp, 80
noise_
qpp::QNoisyEngine, 311
noise_results_
qpp::QNoisyEngine, 311
noise_type
qpp::NoiseBase, 241
NoiseBase
qpp::NoiseBase, 242
none
qpp::Dynamic_bitset, 163
nop
qpp::QCircuit, 291
norm
qpp, 81
normalize
qpp, 81
nq_
qpp::QCircuit, 297
number_theory.h, 389

offset_
qpp::Dynamic_bitset, 164
omega
qpp, 82
one
qpp::States, 338
operations.h, 391
operator!=
qpp::Dynamic_bitset, 164
qpp::QCircuit::iterator, 215
operator<<
qpp::IDisplay, 192
qpp::QCircuit, 294, 295
operator*
qpp::QCircuit::iterator, 216
operator()
qpp::NoiseBase, 245, 246
qpp::internal::EqualEigen, 169
qpp::internal::HashEigen, 189
operator++
qpp::QCircuit::iterator, 216
operator-
qpp::Dynamic_bitset, 164
operator=
qpp::IDisplay, 191
qpp::IJSON, 194
qpp::QCircuit::iterator, 216
qpp::QCircuit::iterator::value_type_, 359
qpp::QEngine, 305
qpp::Timer, 349
qpp::internal::IOManipPointer, 203
qpp::internal::IOManipRange, 206
qpp::internal::Singleton, 331
operator==
qpp::Dynamic_bitset, 165
qpp::QCircuit::iterator, 217
operator"" _bra
qpp::literals, 125
operator"" _i
qpp, 82
qpp::literals, 125
operator"" _ket
qpp::literals, 126
operator"" _prj
qpp::literals, 126

p_
qpp::internal::IOManipPointer, 204
pGHZ
qpp::States, 341

pb00
qpp::States, 340
pb01
qpp::States, 340
pb10
qpp::States, 341
pb11
qpp::States, 341
pi
qpp, 116
plus
qpp::States, 338
pointer
qpp::QCircuit::iterator, 214
powm
qpp, 82
prj
qpp, 83
prng_
qpp::RandomDevices, 329
probs_
qpp::NoiseBase, 247
qpp::QEngine, 307
prod
qpp, 83, 84
psi_
qpp::QEngine, 307
ptrace
qpp, 84, 85
ptrace1
qpp, 85, 86
ptrace2
qpp, 86, 87
ptranspose
qpp, 87, 88
pW
qpp::States, 341
px0
qpp::States, 341
px1
qpp::States, 341
py0
qpp::States, 342
py1
qpp::States, 342
pz0
qpp::States, 342
pz1
qpp::States, 342

QCircuit
qpp::QCircuit, 274
QEngine
qpp::QCircuit, 295
qpp::QEngine, 300, 301
QFT
qpp, 88
qpp::QCircuit, 291, 292
QNoisyEngine
qpp::QNoisyEngine, 309
QPP_UNUSED_
qpp.h, 395
qc_
qpp::QCircuit::iterator, 218
qpp::QEngine, 307
qmutualinfo
qpp, 89
qpp, 13
absm, 29
abssq, 29, 30
adjoint, 30
anticomm, 31
apply, 31–33
applyCTRL, 34
applyQFT, 35
applyTFQ, 36
avg, 36
bigint, 26
bloch2rho, 36
bra, 26
choi2kraus, 37
choi2super, 37
chop, 116
cmat, 27
comm, 38
complement, 38
compperm, 39
concurrence, 39
conjugate, 40
contfrac2x, 40
convergents, 40, 41
cor, 41
cosm, 42
cov, 42
cplx, 27
cwise, 43
det, 43
dirsum, 44, 45
dirsumpow, 45
disp, 46–48
dmat, 27
dyn_col_vect, 27
dyn_mat, 27
dyn_row_vect, 28
ee, 116
egcd, 48
eig, 49
entanglement, 49, 50
entropy, 50, 51
evals, 51
evects, 51
expm, 52
factors, 52
funm, 53
gcd, 53
gconcurrence, 54
grams, 54, 55

hash_eigen, 56
heig, 56
hevals, 57
hevects, 57
idx, 28
infty, 116
inverse, 57
invperm, 58
ip, 58, 59
isprime, 59
ket, 28
kraus2choi, 59
kraus2super, 60
kron, 60–62
kronpow, 62
lcm, 63
load, 64
loadMATLAB, 64, 65
logdet, 66
logm, 66
lognegativity, 67
marginalX, 67
marginalY, 69
maxn, 116
measure, 69–74
measure_seq, 74, 75
mket, 75, 76
modinv, 76
modmul, 77
modpow, 77
mprj, 78
multiidx2n, 79
n2multiidx, 79
negativity, 80
norm, 81
normalize, 81
omega, 82
operator"" _i, 82
pi, 116
powm, 82
prj, 83
prod, 83, 84
ptrace, 84, 85
ptrace1, 85, 86
ptrace2, 86, 87
ptranspose, 87, 88
QFT, 88
qmutualinfo, 89
rand, 90–92
randH, 92
randidx, 93
randket, 93
randkraus, 93
randn, 94, 95
randperm, 96
randprime, 96
randprob, 97
randrho, 97
randU, 97
randV, 98
renyi, 98, 99
reshape, 99
rho2bloch, 100
rho2pure, 100
save, 101
saveMATLAB, 101, 102
schatten, 102
schmidtA, 103
schmidtB, 103, 104
schmidtcoeffs, 104, 105
schmidtprobs, 105, 106
sigma, 106
sinm, 107
spectralpowm, 107
sqrtm, 108
sum, 108, 109
super2choi, 109
svals, 110
svd, 110
svdU, 110
svdV, 111
syspermute, 111, 112
TFQ, 112
to_void, 28
trace, 112
transpose, 113
tsallis, 113, 114
uniform, 114
var, 115
x2contfrac, 115
qpp.h, 393
QPP_UNUSED_, 395
qpp::Bit_circuit, 129
∼Bit_circuit, 132
bCNOT_, 136
bFRED_, 136
bNOT_, 136
bSWAP_, 136
bTOF_, 136
Bit_circuit, 131
btotal_, 136
CNOT, 132
count_, 136
depth_, 137
FRED, 132
get_gate_count, 133
get_gate_depth, 133
NOT, 134
reset, 134
SWAP, 134
TOF, 135
X, 135
qpp::Codes, 137
∼Codes, 139
Codes, 139
codeword, 139

  - internal::Singleton< const Codes >, 139
  - Type, 138
- qpp::Dynamic_bitset, 158
  - ∼Dynamic_bitset, 161
  - all, 161
  - any, 161
  - count, 161
  - data, 161
  - display, 162
  - Dynamic_bitset, 160
  - flip, 162
  - get, 163
  - index_, 163
  - n_, 168
  - none, 163
  - offset_, 164
  - operator!=, 164
  - operator-, 164
  - operator==, 165
  - rand, 165, 166
  - reset, 166
  - set, 166, 167
  - size, 167
  - storage_size, 167
  - storage_size_, 168
  - storage_type, 160
  - to_string, 167
  - v_, 168
  - value_type, 160
- qpp::Gates, 173
  - ∼Gates, 176
  - CNOTba, 183
  - CNOT, 183
  - CTRL, 176
  - CZ, 184
  - expandout, 177, 178
  - FRED, 184
  - Fd, 179
  - Gates, 176
  - get_name, 179
  - H, 184
  - Id, 179
  - Id2, 184
  - internal::Singleton< const Gates >, 183
  - MODMUL, 180
  - Rn, 180
  - RX, 181
  - RY, 181
  - RZ, 182
  - S, 184
  - SWAPd, 182
  - SWAP, 184
  - T, 185
  - TOF, 185
  - X, 185
  - Xd, 182
  - Y, 185
  - Z, 185
  - Zd, 183
- qpp::IDisplay, 189
  - ∼IDisplay, 191
  - display, 191
  - IDisplay, 190, 191
  - operator<<, 192
  - operator=, 191
- qpp::IJSON, 192
  - ∼IJSON, 193
  - IJSON, 193
  - operator=, 194
  - to_JSON, 194
- qpp::Init, 195
  - ∼Init, 196
  - Init, 196
  - internal::Singleton< const Init >, 196
- qpp::NoiseBase
  - ∼NoiseBase, 242
  - compute_probs_, 243
  - compute_state_, 243
  - d_, 246
  - generated_, 246
  - get_Ks, 244
  - get_d, 243
  - get_last_idx, 244
  - get_last_K, 244
  - get_last_p, 244
  - get_probs, 245
  - i_, 247
  - Ks_, 247
  - noise_type, 241
  - NoiseBase, 242
  - operator(), 245, 246
  - probs_, 247
- qpp::NoiseBase< T >, 240
- qpp::NoiseType, 247
- qpp::NoiseType::StateDependent, 334
- qpp::NoiseType::StateIndependent, 334
- qpp::QCircuit, 268
  - ∼QCircuit, 275
  - add_hash_, 275
  - begin, 275
  - cCTRL_custom, 278
  - cCTRL, 276, 277
  - CTRL_custom, 281
  - CTRL, 279, 280
  - cbegin, 276
  - cend, 278
  - cmat_hash_tbl_, 295
  - const_iterator, 273
  - count_, 296
  - d_, 296
  - display, 281
  - end, 281, 282
  - gate, 282, 283
  - gate_custom, 283
  - gate_fan, 284, 285
  - GateType, 273

gates_, 296
get_cmat_hash_tbl_, 285
get_d, 285
get_gate_count, 285
get_gate_depth, 286
get_gates_, 286
get_measured, 286, 287
get_measurement_count, 287
get_measurements_, 288
get_name, 288
get_nc, 288
get_non_measured, 288
get_nop_count, 289
get_nq, 289
get_step_count, 289
MeasureType, 274
measured_, 296
measurement_count_, 296
measurements_, 296
measureV, 289, 290
measureZ, 290
name_, 297
nc_, 297
nop, 291
nq_, 297
operator<<, 294, 295
QCircuit, 274
QEngine, 295
QFT, 291, 292
step_types_, 297
StepType, 274
TFQ, 292, 293
to_JSON, 293
qpp::QCircuit::GateStep, 186
ctrl_, 187
gate_hash_, 187
gate_type_, 188
GateStep, 187
name_, 188
target_, 188
qpp::QCircuit::MeasureStep, 235
c_reg_, 237
mats_hash_, 237
MeasureStep, 236
measurement_type_, 237
name_, 237
target_, 237
qpp::QCircuit::iterator, 213
difference_type, 214
elem_, 218
iterator, 215
iterator_category, 214
operator!=, 215
operator∗, 216
operator++, 216
operator=, 216
operator==, 217
pointer, 214
qc_, 218
reference, 215
set_begin_, 217
set_end_, 217
value_type, 215
qpp::QCircuit::iterator::value_type_, 357
display, 359
gates_ip_, 359
ip_, 360
measurements_ip_, 360
operator=, 359
type_, 360
value_type_, 358, 359
value_type_qc_, 360
qpp::QEngine, 298
∼QEngine, 301
display, 301
dits_, 307
execute, 301, 302
get_circuit, 302
get_dit, 302
get_dits, 303
get_measured, 303
get_non_measured, 303
get_probs, 304
get_psi, 304
get_relative_pos_, 304
operator=, 305
probs_, 307
psi_, 307
QEngine, 300, 301
qc_, 307
reset, 305
set_dit, 305
set_measured_, 305
set_psi, 306
subsys_, 307
to_JSON, 306
qpp::QNoisyEngine
execute, 310, 311
get_noise_results, 311
noise_, 311
noise_results_, 311
QNoisyEngine, 309
qpp::QNoisyEngine< NoiseModel >, 308
qpp::QubitAmplitudeDampingNoise, 312
QubitAmplitudeDampingNoise, 313
qpp::QubitBitFlipNoise, 313
QubitBitFlipNoise, 315
qpp::QubitBitPhaseFlipNoise, 315
QubitBitPhaseFlipNoise, 316
qpp::QubitDepolarizingNoise, 317
QubitDepolarizingNoise, 318
qpp::QubitPhaseDampingNoise, 318
QubitPhaseDampingNoise, 319
qpp::QubitPhaseFlipNoise, 320
QubitPhaseFlipNoise, 321
qpp::QuditDepolarizingNoise, 323

fill_Ks_, 325
fill_probs_, 325
QuditDepolarizingNoise, 324
qpp::RandomDevices, 326
∼RandomDevices, 327
get_prng, 328
internal::Singleton< RandomDevices >, 329
load, 328
prng_, 329
RandomDevices, 327
rd_, 329
save, 328
qpp::States, 334
∼States, 337
b00, 339
b01, 340
b10, 340
b11, 340
GHZ, 340
internal::Singleton< const States >, 339
jn, 337
mes, 337
minus, 338
one, 338
pGHZ, 341
pb00, 340
pb01, 340
pb10, 341
pb11, 341
plus, 338
pW, 341
px0, 341
px1, 341
py0, 342
py1, 342
pz0, 342
pz1, 342
States, 337
W, 342
x0, 342
x1, 343
y0, 343
y1, 343
z0, 343
z1, 343
zero, 339
qpp::Timer
∼Timer, 348
display, 348
end_, 350
get_duration, 349
operator=, 349
start_, 350
tic, 349
tics, 350
Timer, 347, 348
toc, 350
qpp::Timer< T, CLOCK_T >, 346
qpp::exception, 116
qpp::exception::CustomException, 140
CustomException, 141
type_description, 142
what_, 142
qpp::exception::DimsInvalid, 143
Exception, 144
type_description, 144
qpp::exception::DimsMismatchCvector, 145
Exception, 146
type_description, 146
qpp::exception::DimsMismatchMatrix, 147
Exception, 148
type_description, 148
qpp::exception::DimsMismatchRvector, 149
Exception, 150
type_description, 150
qpp::exception::DimsMismatchVector, 151
Exception, 152
type_description, 152
qpp::exception::DimsNotEqual, 153
Exception, 154
type_description, 154
qpp::exception::Duplicates, 156
Exception, 157
type_description, 157
qpp::exception::Exception, 170
Exception, 172
msg_, 173
type_description, 172
what, 172
where_, 173
qpp::exception::InvalidIterator, 197
Exception, 198
type_description, 198
qpp::exception::MatrixMismatchSubsys, 219
Exception, 220
type_description, 221
qpp::exception::MatrixNotCvector, 221
Exception, 222
type_description, 223
qpp::exception::MatrixNotRvector, 223
Exception, 224
type_description, 225
qpp::exception::MatrixNotSquare, 225
Exception, 226
type_description, 227
qpp::exception::MatrixNotSquareNorCvector, 227
Exception, 228
type_description, 229
qpp::exception::MatrixNotSquareNorRvector, 229
Exception, 230
type_description, 231
qpp::exception::MatrixNotSquareNorVector, 231
Exception, 232
type_description, 233
qpp::exception::MatrixNotVector, 233
Exception, 234

type_description, 235
qpp::exception::NoCodeword, 238
Exception, 239
type_description, 239
qpp::exception::NotBipartite, 248
Exception, 249
type_description, 250
qpp::exception::NotImplemented, 250
Exception, 251
type_description, 252
qpp::exception::NotQubitCvector, 252
Exception, 253
type_description, 254
qpp::exception::NotQubitMatrix, 254
Exception, 255
type_description, 256
qpp::exception::NotQubitRvector, 256
Exception, 257
type_description, 258
qpp::exception::NotQubitSubsys, 258
Exception, 259
type_description, 260
qpp::exception::NotQubitVector, 260
Exception, 261
type_description, 262
qpp::exception::OutOfRange, 262
Exception, 263
type_description, 264
qpp::exception::PermInvalid, 264
Exception, 265
type_description, 266
qpp::exception::PermMismatchDims, 266
Exception, 267
type_description, 268
qpp::exception::QuditAlreadyMeasured, 321
Exception, 322
type_description, 323
qpp::exception::SizeMismatch, 332
Exception, 333
type_description, 333
qpp::exception::SubsysMismatchDims, 344
Exception, 345
type_description, 345
qpp::exception::TypeMismatch, 351
Exception, 352
type_description, 353
qpp::exception::UndefinedType, 353
Exception, 354
type_description, 355
qpp::exception::Unknown, 355
Exception, 356
type_description, 357
qpp::exception::ZeroSize, 361
Exception, 362
type_description, 362
qpp::experimental, 118
qpp::internal, 118
check_cvector, 120
check_dims, 120
check_dims_match_cvect, 120
check_dims_match_mat, 120
check_dims_match_rvect, 120
check_eq_dims, 121
check_matching_sizes, 121
check_no_duplicates, 121
check_nonzero_size, 121
check_perm, 121
check_qubit_cvector, 121
check_qubit_matrix, 122
check_qubit_rvector, 122
check_qubit_vector, 122
check_rvector, 122
check_square_mat, 122
check_subsys_match_dims, 122
check_vector, 123
dirsum2, 123
get_dim_subsys, 123
get_num_subsys, 123
hash_combine, 123
kron2, 123
multiidx2n, 124
n2multiidx, 124
variadic_vector_emplace, 124
qpp::internal::Display_Impl_, 155
display_impl_, 155
qpp::internal::EqualEigen, 169
operator(), 169
qpp::internal::HashEigen, 188
operator(), 189
qpp::internal::IOManipEigen, 199
A_, 200
chop_, 201
display, 200
IOManipEigen, 200
qpp::internal::IOManipPointer
display, 203
end_, 203
IOManipPointer, 202, 203
N_, 204
operator=, 203
p_, 204
separator_, 204
start_, 204
qpp::internal::IOManipPointer< PointerType >, 201
qpp::internal::IOManipRange
display, 206
end_, 207
first_, 207
IOManipRange, 206
last_, 207
operator=, 206
separator_, 207
start_, 207
qpp::internal::IOManipRange< InputIterator >, 205
qpp::internal::Singleton
~Singleton, 331

get_instance, 331
get_thread_local_instance, 331
operator=, 331
Singleton, 330, 331
qpp::internal::Singleton< T >, 329
qpp::is_complex< std::complex< T > >, 209
qpp::is_complex< T >, 208
qpp::is_iterable< T, to_void< decltype(std::declval< T >().begin()), decltype(std::declval< T >().↩ end()), decltype(∗(std::declval< T >().↩ begin()))> >, 211
qpp::is_iterable< T, typename >, 210
qpp::is_matrix_expression< Derived >, 212
qpp::literals, 125
operator"" _bra, 125
operator"" _i, 125
operator"" _ket, 126
operator"" _prj, 126
qpp::make_void
type, 219
qpp::make_void< Ts >, 218
QubitAmplitudeDampingNoise
qpp::QubitAmplitudeDampingNoise, 313
QubitBitFlipNoise
qpp::QubitBitFlipNoise, 315
QubitBitPhaseFlipNoise
qpp::QubitBitPhaseFlipNoise, 316
QubitDepolarizingNoise
qpp::QubitDepolarizingNoise, 318
QubitPhaseDampingNoise
qpp::QubitPhaseDampingNoise, 319
QubitPhaseFlipNoise
qpp::QubitPhaseFlipNoise, 321
QuditDepolarizingNoise
qpp::QuditDepolarizingNoise, 324

rand
qpp, 90–92
qpp::Dynamic_bitset, 165, 166
randH
qpp, 92
randidx
qpp, 93
randket
qpp, 93
randkraus
qpp, 93
randn
qpp, 94, 95
random.h, 395
RandomDevices
qpp::RandomDevices, 327
randperm
qpp, 96
randprime
qpp, 96
randprob
qpp, 97
randrho
qpp, 97
randU
qpp, 97
randV
qpp, 98
rd_
qpp::RandomDevices, 329
reference
qpp::QCircuit::iterator, 215
renyi
qpp, 98, 99
reset
qpp::Bit_circuit, 134
qpp::Dynamic_bitset, 166
qpp::QEngine, 305
reshape
qpp, 99
rho2bloch
qpp, 100
rho2pure
qpp, 100
Rn
qpp::Gates, 180
RX
qpp::Gates, 181
RY
qpp::Gates, 181
RZ
qpp::Gates, 182

S
qpp::Gates, 184
SWAPd
qpp::Gates, 182
SWAP
qpp::Bit_circuit, 134
qpp::Gates, 184
save
qpp, 101
qpp::RandomDevices, 328
saveMATLAB
qpp, 101, 102
schatten
qpp, 102
schmidtA
qpp, 103
schmidtB
qpp, 103, 104
schmidtcoeffs
qpp, 104, 105
schmidtprobs
qpp, 105, 106
separator_
qpp::internal::IOManipPointer, 204
qpp::internal::IOManipRange, 207
set
qpp::Dynamic_bitset, 166, 167
set_begin_
qpp::QCircuit::iterator, 217

set_dit
qpp::QEngine, 305
set_end_
qpp::QCircuit::iterator, 217
set_measured_
qpp::QEngine, 305
set_psi
qpp::QEngine, 306
sigma
qpp, 106
Singleton
qpp::internal::Singleton, 330, 331
sinm
qpp, 107
size
qpp::Dynamic_bitset, 167
spectralpowm
qpp, 107
sqrtm
qpp, 108
start_
qpp::Timer, 350
qpp::internal::IOManipPointer, 204
qpp::internal::IOManipRange, 207
States
qpp::States, 337
statistics.h, 397
step_types_
qpp::QCircuit, 297
StepType
qpp::QCircuit, 274
storage_size
qpp::Dynamic_bitset, 167
storage_size_
qpp::Dynamic_bitset, 168
storage_type
qpp::Dynamic_bitset, 160
subsys_
qpp::QEngine, 307
sum
qpp, 108, 109
super2choi
qpp, 109
svals
qpp, 110
svd
qpp, 110
svdU
qpp, 110
svdV
qpp, 111
syspermute
qpp, 111, 112

T
qpp::Gates, 185
TFQ
qpp, 112
qpp::QCircuit, 292, 293
TOF
qpp::Bit_circuit, 135
qpp::Gates, 185
target_
qpp::QCircuit::GateStep, 188
qpp::QCircuit::MeasureStep, 237
tic
qpp::Timer, 349
tics
qpp::Timer, 350
Timer
qpp::Timer, 347, 348
to_JSON
qpp::IJSON, 194
qpp::QCircuit, 293
qpp::QEngine, 306
to_string
qpp::Dynamic_bitset, 167
to_void
qpp, 28
toc
qpp::Timer, 350
trace
qpp, 112
traits.h, 398
transpose
qpp, 113
tsallis
qpp, 113, 114
Type
qpp::Codes, 138
type
qpp::make_void, 219
type_
qpp::QCircuit::iterator::value_type_, 360
type_description
qpp::exception::CustomException, 142
qpp::exception::DimsInvalid, 144
qpp::exception::DimsMismatchCvector, 146
qpp::exception::DimsMismatchMatrix, 148
qpp::exception::DimsMismatchRvector, 150
qpp::exception::DimsMismatchVector, 152
qpp::exception::DimsNotEqual, 154
qpp::exception::Duplicates, 157
qpp::exception::Exception, 172
qpp::exception::InvalidIterator, 198
qpp::exception::MatrixMismatchSubsys, 221
qpp::exception::MatrixNotCvector, 223
qpp::exception::MatrixNotRvector, 225
qpp::exception::MatrixNotSquare, 227
qpp::exception::MatrixNotSquareNorCvector, 229
qpp::exception::MatrixNotSquareNorRvector, 231
qpp::exception::MatrixNotSquareNorVector, 233
qpp::exception::MatrixNotVector, 235
qpp::exception::NoCodeword, 239
qpp::exception::NotBipartite, 250
qpp::exception::NotImplemented, 252
qpp::exception::NotQubitCvector, 254

qpp::exception::NotQubitMatrix, 256
qpp::exception::NotQubitRvector, 258
qpp::exception::NotQubitSubsys, 260
qpp::exception::NotQubitVector, 262
qpp::exception::OutOfRange, 264
qpp::exception::PermInvalid, 266
qpp::exception::PermMismatchDims, 268
qpp::exception::QuditAlreadyMeasured, 323
qpp::exception::SizeMismatch, 333
qpp::exception::SubsysMismatchDims, 345
qpp::exception::TypeMismatch, 353
qpp::exception::UndefinedType, 355
qpp::exception::Unknown, 357
qpp::exception::ZeroSize, 362
types.h, 399

uniform
qpp, 114

v_
qpp::Dynamic_bitset, 168
value_type
qpp::Dynamic_bitset, 160
qpp::QCircuit::iterator, 215
value_type_
qpp::QCircuit::iterator::value_type_, 358, 359
value_type_qc_
qpp::QCircuit::iterator::value_type_, 360
var
qpp, 115
variadic_vector_emplace
qpp::internal, 124

W
qpp::States, 342
what
qpp::exception::Exception, 172
what_
qpp::exception::CustomException, 142
where_
qpp::exception::Exception, 173

X
qpp::Bit_circuit, 135
qpp::Gates, 185
x0
qpp::States, 342
x1
qpp::States, 343
x2contfrac
qpp, 115
Xd
qpp::Gates, 182

Y
qpp::Gates, 185
y0
qpp::States, 343
y1
qpp::States, 343

Z
qpp::Gates, 185
z0
qpp::States, 343
z1
qpp::States, 343
Zd
qpp::Gates, 183
zero
qpp::States, 339