Quantum Resource Estimator Documentation

Overview

This project is a web-based tool designed to estimate the physical resources required for fault-tolerant quantum computing algorithms. It helps users understand the overhead involved in running quantum circuits on error-corrected hardware, accounting for factors like error rates, gate counts, and different error correction schemes. The tool is built using JavaScript and assumes a basic HTML structure for the user interface (e.g., input fields, dropdowns, and display areas), though the core logic is in the provided code.

The estimator focuses on key quantum computing concepts:

- **Logical qubits**: Error-protected qubits that represent the algorithm's qubits.
- **Physical qubits**: The actual hardware qubits needed, including overhead for error correction.
- **Gates and operations**: Different types (T gates, Clifford gates, rotations, measurements) with varying costs.
- **Error correction**: Uses schemes like surface codes to suppress errors to a target rate.
- **Hardware platforms**: Influences gate times and thus runtime.

The tool supports preset algorithms (e.g., Shor's for factoring, Grover's search) or custom inputs. It calculates resources for multiple error correction schemes, displays results in a user-friendly format, and provides insights. Additional features include exporting results to PDF and sharing via URL.

This documentation explains the code structure, key functions, calculations, and underlying quantum concepts in a step-by-step manner to make it accessible for beginners while providing depth for advanced users.

Key Components and Data Structures

Presets

The tool includes predefined algorithm configurations stored in the `presets` object. These represent realistic estimates for common quantum algorithms:

- `shor-2048` : Simulates Shor's algorithm for factoring a 2048-bit number.
  - Logical qubits: 4096
  - T gates: 2,000,000,000
  - Clifford gates: 10,000,000,000
  - Rotation gates: 100,000,000
  - Measurements: 50,000,000
  - Target error rate: 0.0001 (0.01%)
- `grover-256` : Grover's search on a 256-bit space (e.g., for cryptographic attacks).
  - Logical qubits: 256
  - T gates: 500,000,000
  - Clifford gates: 2,000,000,000
  - Rotation gates: 50,000,000
  - Measurements: 10,000,000
  - Target error rate: 0.001 (0.1%)
- `quantum-sim` : A medium-scale quantum simulation (e.g., for many-body physics).
  - Logical qubits: 50
  - T gates: 10,000,000
  - Clifford gates: 50,000,000
  - Rotation gates: 5,000,000
  - Measurements: 1,000,000
  - Target error rate: 0.001 (0.1%)

- `vqe-small` : Variational Quantum Eigensolver for a small molecule.
    - Logical qubits: 20
    - T gates: 50,000
    - Clifford gates: 200,000
    - Rotation gates: 20,000
    - Measurements: 10,000
    - Target error rate: 0.01 (1%)

These presets can be loaded via a dropdown (ID: 'algorithmPreset'), which populates input fields. Users can select 'custom' for manual entry.

Constants and Assumptions

- `tPerRotation` : Number of T gates needed to approximate one rotation gate (for precision ~10^{-8}). Calculated as `Math.ceil(3 * Math.log2(1 / 1e-8))` , typically around 50–70. This uses the Solovay-Kitaev theorem approximation.
- **Physical error rate**: Fixed at 0.001 (0.1%), a realistic near-term value for hardware like superconducting qubits.
- **Threshold error rate**: 0.01 (1%), the point below which surface codes can suppress errors effectively.
- **Gate times**: Hardware-specific (in seconds):
  - Superconducting: 50 ns
  - Ion: 5 µs
  - Photonic: 1 ns
  - Neutral: 2 µs
- **Error correction schemes**: An array of objects, each defining a scheme (e.g., Surface Code, Cat Qubits). Properties include:
  - `physPerLog(d)` : Physical qubits per logical qubit as a function of code distance `d` .
  - `tFactorySize` : Qubits per T-state factory (e.g., 180 for Surface Code).
  - `distillationRounds` : Rounds of magic state distillation (1–2).
  - `cliffordCyclesPerGate` , `tCyclesPerGate(d)` , `measurementCycles(d)` : Cycles needed per operation type.
  - `feasibility` : 'near-term' or 'long-term'.
  - `description` : Brief overview.

User Interface and Event Handling

The tool interacts with HTML elements via `document.getElementById` . Key functions:

- `loadPreset()` : Loads a preset into input fields and triggers calculation. Called on preset dropdown change.
- `setQuickValue(id, value)` : Sets an input field value and recalculates. Useful for quick tests.
- `calculateResources()` : Core function (detailed below). Triggered on input changes or preset loads.
- `displayResults(results)` : Renders summary stats, scheme cards, and insights in the UI.
- `selectScheme(idx)` : Highlights a scheme card on click.
- `exportPDF()` : Generates a printable HTML report and opens a print window.
- `shareResults()` : Creates a shareable URL with parameters and copies to clipboard.
- `showInfo()` : Displays a help alert with key concepts.
- `showToast(message)` : Shows temporary notifications.
- `formatNumber(n)` : Adds commas to large numbers.
- `formatTime(s)` : Converts seconds to human-readable units (ns, μs, ms, s, min, hrs, days, years).

Event listeners auto-calculate on changes. URL parameters (e.g., ?qubits=50) preload values.

Core Calculation Logic: calculateResources()

This function computes resources based on user inputs. It handles edge cases and uses quantum error correction theory.

Step 1: Parse Inputs

- Logical qubits, T gates, Clifford gates, rotation gates, measurements: Integers (default 0).
- Error rate: Float (e.g., 0.001).
- Hardware: String (e.g., 'superconducting').

If logical qubits = 0 or no operations, show toast and exit.

Step 2: Total Operations and Error Budget

- `totalLogicalOps` : Sum of non-zero gate/measurement counts.
- `perOpErrorBudget` : Error rate divided by total operations (distributes error budget evenly).

Step 3: Code Distance Calculation

Code distance `d` determines error suppression. Uses surface code formula:

- Logical error rate ≈ 0.1 × (physical_error / threshold)^{(d+1)/2}
- Solve for `d` : `d = 2 * log(perOpErrorBudget / 0.1) / log(physical_error / threshold) - 1`
- Round up, ensure odd, min 3, max 51.

This ensures the per-operation logical error meets the budget.

Step 4: Effective T Gates and Depth

- `effectiveTGates` : T gates + (rotations × tPerRotation), as rotations decompose into T gates.
- `parallelismFactor` : ≈ √(logical qubits), assuming some parallel T gate execution.
- `tDepth` : effectiveTGates / parallelismFactor (ceil).
- `cliffordDepth` : cliffordGates / logicalQubits (ceil, better parallelism).
- `measurementRounds` : measurements / logicalQubits (ceil).

Step 5: Scheme-Specific Calculations

Loop over schemes:

- `physPerLog` : Physical qubits per logical (e.g., $2d^2$ for Surface Code).
- `dataQubits` : logicalQubits × physPerLog.
- T-factory qubits (if effectiveTGates > 0):
  - Cycles per T-state: d × 15^distillationRounds (15-to-1 protocol).
  - T-states per second: 1 / (cyclesPerTState × codeCycleTime), where codeCycleTime = d × gateTime.
  - Circuit time: Sum of Clifford and T gate times (depth × cycles per gate × codeCycleTime).
  - Required T rate: effectiveTGates / circuitTime.
  - `factoryCount` : Ceil(required rate / production rate).
  - `tFactoryQubits` : factoryCount × (tFactorySize × distillationRounds).
- Total `physicalQubits` : dataQubits + tFactoryQubits.
- Total cycles: Sum of T, Clifford, and measurement cycle depths (using scheme-specific multipliers).
- `runtime` : totalCycles × codeCycleTime.
- Success probability: 1 - (totalCycles × logicalErrorPerCycle), capped 0–100%. Logical error per cycle: 0.1 × (physical_error / threshold)^{(d+1)/2}.

Results stored in `currentResults` for display/export.

Error Correction Schemes Explained

Each scheme models a quantum error correction code:

1. **Surface Code**:
    - Physical per logical: $2d^2$ (data + ancilla qubits in a 2D lattice).
    - T-factory: 180 qubits/factory, 2 distillation rounds.
    - Cycles: 1 for Clifford, d for T/measurement.
    - Near-term, mature, requires 2D nearest-neighbor connections.
    - Pros: Simple, well-studied. Cons: High qubit overhead.
2. **Cat Qubits**:
    - Physical per logical: 6d (bosonic code, linear scaling in one direction).
    - T-factory: 90 qubits/factory, 2 rounds.
    - Similar cycles to surface code.
    - Near-term, lower overhead for biased noise.
    - Pros: Efficient for certain hardware (e.g., photons). Cons: Experimental.
3. **Color Code**:
    - Physical per logical: $2.5d^2$.
    - T-factory: 120 qubits/factory, 1 round (transversal gates help).
    - Cycles: 0.5 for Clifford (transversal), 0.8d for T.
    - Long-term, transversal Cliffords.
    - Pros: Faster Cliffords. Cons: Complex lattice.
4. **Bacon-Shor**:
    - Physical per logical: $1.5d^2$ (gauge freedom reduces count).
    - T-factory: 150 qubits/factory, 2 rounds.
    - Cycles: 1.2 for Clifford, 1.1d for T.
    - Long-term, good for asymmetric errors.
    - Pros: Lower qubits. Cons: Higher cycle overhead.

These are approximations; real implementations vary.

Displaying Results

- **Summary**: Shows best scheme's qubits, distance, runtime.
- **Scheme Cards**: Grid of cards with metrics, colored for feasibility (green/warning/red).
- **Insights**: List of key takeaways, e.g., T-gate dominance, hardware feasibility, optimization tips.

Export and Sharing

- **PDF Export**: Creates a styled HTML report with parameters, recommended config, comparison table, gate analysis, cycle breakdown. Opens in a new window for printing.
- **Share**: Encodes params in URL (e.g., ?qubits=50&tgates=10000000).

Underlying Quantum Concepts

- **Fault Tolerance**: Quantum hardware is noisy; error correction codes (like surface codes) use many physical qubits to encode one logical qubit, suppressing errors exponentially with distance d.
- **Gate Types**:
  - Clifford: Easy (transversal in many codes).
  - T: Hard (requires "magic states" distilled in factories).
  - Rotations: Approximated by sequences of T and Clifford gates.
- **Magic State Distillation**: T gates need pure ancillary states; factories use 15 noisy states to produce 1 good one, repeated for fidelity.
- **Runtime**: Depends on circuit depth (parallel gates) and code cycles (error checks every ~d gates).
- **Success Probability**: Assumes independent errors; total failure chance grows with cycles.

This tool uses simplified models for education. For production, consider advanced simulators accounting for routing, noise models, and optimizations.

If you have questions or need code modifications, refer to the functions above!