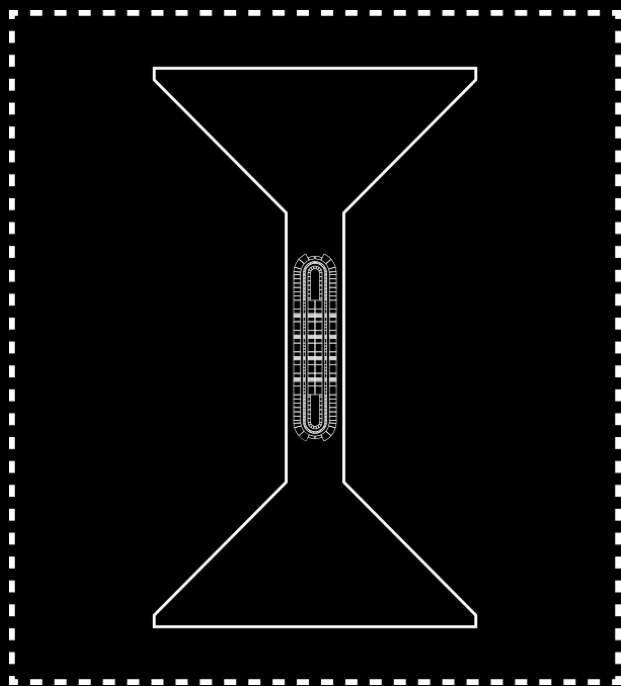


QUANTINUUM



# Quantinuuum System Model H2 Emulator

Product Data Sheet

PRE-RELEASE

Version 0.1 June 8, 2023

■ **TABLE OF CONTENTS**

**PRE-RELEASE ..... 3**

**INTRODUCTION ..... 3**

**FEATURES ..... 3**

**USE CASES..... 3**

**FUNCTIONAL REQUIREMENTS..... 3**

**EMULATOR ACCESS AND OUTPUT ..... 4**

**PERFORMANCE..... 4**

**EMULATION METHOD ..... 5**

**NOISE MODEL ..... 5**

Physical Noise ..... 6

Dephasing Noise..... 7

Arbitrary Angle Noise Scaling..... 8

Scaling..... 9

**APPENDIX ..... 9**

## ■ PRE-RELEASE

This version of the Quantinuum System Model H2 Emulator Data Sheet is in PRE-RELEASE form. Changes may be made to the Data Sheet prior to releasing the Data Sheet publicly.

## ■ INTRODUCTION

This Product Data Sheet covers all features and characteristics of the **Quantinuum System Model H2 Emulator**. This includes specifications for all emulators available online, which currently includes the H2-1 emulator.

## ■ FEATURES

- Detailed noise models and up-to-date parameters closely mimicking System Model H2 (hardware) performance. Each emulator uses the same physical noise model, but noise parameters reflect performance of the device being emulated.
- Uses identical API for job submission as System Model H2, enabling seamless translation from emulator to hardware
- Uses identical compiler as System Model H2, containing all the native gates (single-qubit rotations, two-qubit ZZ-gates), transport operations and classical operations used in System Model H2
- Provides identical output format as System Model H2
- Allows usage of System Model H2 attributes: all-to-all connectivity and qubit reuse after mid-circuit measurement
- Available even while System Model H2 is offline to enable maximized productivity and development time

## ■ USE CASES

The System Model H2 Emulator provides a high-fidelity emulation of System Model H2. Use cases include:

- Debugging of quantum code before running on physical hardware
- Optimization of quantum code in the presence of noise mechanisms
- Exploring new algorithms and techniques for quantum error correction
- Introduction to System Model H2 and its unique differentiating capabilities such as qubit reuse after mid-circuit measurement, all-to-all connectivity, and high-fidelity gates

## ■ FUNCTIONAL REQUIREMENTS

The System Model H2 Emulator is meant to be a functional emulation of System Model H2 and therefore supports the same functional operations as H2. Specifically, the System Model H2 Emulator supports:

- OPENQASM 2.0 circuits
- Quantinuum QASM enhancements, including classical logic, math, and program flow control
- System Model H2 native gate set<sup>1</sup> –  $R_z(\lambda)$ ,  $U_{1q}(\theta, \varphi)$ ,  $ZZ$ ,  $RZZ(\theta)$

---

<sup>1</sup> For definition of native gates, please request a copy of the *Quantinuum System Model H2 Product Data Sheet*

- Common compound gates from OPENQASM library, e.g., CX, H
- User-defined compound gates
- User option of noiseless simulation or inclusion of System Model H2 noise models
- Large quantum circuits with a limit of 10,000 on the number of shots
- Identical queuing prioritization as System Model H2

## ■ EMULATOR ACCESS AND OUTPUT

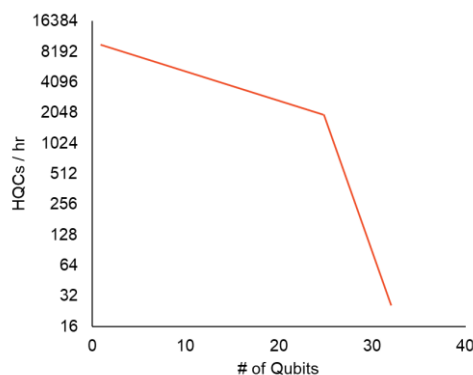
Communication with the System Model H2 Emulator occurs through an API endpoint based on the OpenQASM 2.0 standard [1]. Interface details are given in the *Quantinuum Application Programming Interface (API) Specification*.

Users can select a System Model H2 Emulator in the machine list API, designated with the “E” suffix machine name. The output of the emulator is a JSON-formatted array, identical to the output format of System Model H2. Through the Job Submission API, users may select the type of emulator used and application of error models.

## ■ PERFORMANCE

The performance of the System Model H2 Emulator is measured in comparison to the H2 hardware. With the inclusion of an accurate and up-to-date noise models, the System Model H2 Emulator provides a detailed representation of System Model H2 output, operating on a GPU backend. Because of the memory requirements, a slowdown in the number of H-System Quantum Credits (HQC) processed in an hour is observed for emulations using 25 qubits or more. The difference in processing time is due to the higher number of qubits combined with the realistic physical noise model of System Model H2. To estimate the amount of time the emulation will take, users can use the guideline in Figure 1 in conjunction with the total HQCs for their quantum program.

Accuracy is verified at Quantinuum by comparison between the emulator and hardware outputs. Users should be aware that H2 is a more complex trapping geometry than H1. In comparing the H2 emulation to H2 QPU results, users may experience a larger discrepancy than experienced comparing the H1 emulator to the H1 QPU. Quantinuum expects that as done with H1, error models for H2 will mature over time. In the case of exceptional or unexplained variance, users should contact Quantinuum technical support at [QCsupport@quantinuum.com](mailto:QCsupport@quantinuum.com) to discuss the circuit and results.



**Figure 1 System Model H2 Emulator Average HQCs/Hour**

## ■ EMULATION METHOD

The System Model H2 Emulator, accessible via the API, receives instructions directly from the same compilers used by the System Model H2 physical hardware. These compilers translate the submitted quantum program into a set of instructions comprising the native gate operations and the transport operations necessary to reconfigure the ion chain at each step of the program. Users can choose between either a state-vector or stabilizer emulation method; in both cases results are performed shot-by-shot. The state-vector emulation method can run any general quantum circuits, while the stabilizer emulation method is restricted to circuits involving only quantum unitary gates that are Clifford operations.

The error model for the emulation can be turned on or off, allowing noisy or noise-free emulations, respectively. The emulated error model includes:

- depolarizing gate noise
- leakage errors, crosstalk noise
- dephasing noise due to transport and qubit idling

Except for dephasing, errors on physical qubits are modeled as stochastic processes. For the state-vector emulation, dephasing is handled as a coherent Z rotation according to a dephasing rate and the duration the qubit spends in transport or while idling while other qubits are being gated. For the stabilizer emulation, the dephasing noise is treated as a stochastic Z error where the probability of a Z error is equal to the Pauli twirled approximation of the coherent dephasing channel, which is proportional to the square of the dephasing rate multiplied by the duration.

## ■ NOISE MODEL

Users who have direct access to the Quantinuum API have the option of experimenting with the physical noise parameters of the emulator. When deviating from the default emulation model, users should not assume that performance predicted with modified error parameters will match hardware performance.

All parameters listed in Table 1 are the default settings of the System Model H2 emulators. As updates to the System Model H2 quantum computers are made, the emulator noise parameters and the underlying error model are subject to change to accommodate performance improvements, updates in the methodology for measuring devices parameter and research into the noise sources themselves.

All the errors are applied even when only certain parameters are specified. Only the parameters specified are overridden. To turn off certain error parameters, explicitly set them to 0.

For more information on the errors observed, see the following publications: [Realization of Real-Time Fault-Tolerant Quantum Error Correction](#), [Implementing Fault-tolerant Entangling Gates on the Five-qubit Code and the Color Code](#).

*Table 1 Default Settings of the System Model H2 Emulators*

Default Settings	H2-1
<b>General</b>	
Qubits	32
Connectivity	All-to-all
Parallel two-qubit operations	4
<b>Physical Noise</b>	
Single-Qubit Fault Probability (p1)	$2.5 \times 10^{-5}$
Two-Qubit Fault Probability (p2)	$2 \times 10^{-3}$
Bit Flip Measurement Probability (0 outcome) (p_meas)	$0.82 \times 10^{-3}$
Bit Flip Measurement Probability (1 outcome) (p_meas)	$2.2 \times 10^{-3}$
Crosstalk Measurement Fault Probability (p_crosstalk_meas)	$1.3 \times 10^{-5}$
Initialization Fault Probability (p_init)	$4 \times 10^{-5}$
Crosstalk Initialization Probability (p_crosstalk_init)	$9.6 \times 10^{-6}$
Ratio of Single-Qubit Spontaneous Emission to p1 (p1_emission_ratio)	0.32
Ratio of Single-Qubit Spontaneous Emission in Two-Qubit Gate to p2 (p2_emission_ratio)	0.395
<b>Dephasing Noise</b>	
Quadratic Dephasing Rate (quadratic_dephasing_rate)	0.043
Linear Dephasing Rate (linear_dephasing_rate)	0.0028
Coherent to Incoherent Factor (coherent_to_incoherent_factor)	2.0
<b>Arbitrary Angle Noise Scaling</b>	
Fit Parameter 1 (przz_a)	1.39
Fit Parameter 2 (przz_b)	0.014
Fit Parameter 3 (przz_c)	1.39
Fit Parameter 4 (przz_d)	0.014

## Physical Noise

The emulator runs with default error parameters that represent a noise environment that closely resembles the respective hardware. These error parameters can be set and used to override the default error parameters and do finer-grain tweaks of the error model. Modification of the error parameters away from default values is an advanced option and not recommended as a starting point for emulations of hardware performance.

- **Single-Qubit Fault Probability (p1):** probability of a fault occurring during a single-qubit gate
- **Two-Qubit Fault Probability (p2):** probability of a fault occurring during a two-qubit gate

- **Bit Flip Measurement Probability ( $p_{\text{meas}}$ ):** probability of a bit flip being applied to a measurement. Either a float or a tuple of 2 floats. If it is a single float then that error rate is used to bitflip both 0 and 1 measurement results. If a tuple is supplied, the first element is the probability a bit flip is applied if a 0 result occurs during measurement while the second error rate if a 1 is measured.
- **Crosstalk Measurement Fault Probability ( $p_{\text{crosstalk\_meas}}$ ):** probability of a crosstalk measurement fault occurring
- **Initialization Fault Probability ( $p_{\text{init}}$ ):** probability of a fault occurring during initialization of a qubit
- **Crosstalk Initialization Fault Probability ( $p_{\text{crosstalk\_init}}$ ):** probability of a cross-talk fault occurring during initialization of a qubit
- **Ratio of Single-Qubit Spontaneous Emission to  $p_1$  ( $p_1_{\text{emission\_ratio}}$ ):** fraction of  $p_1$  that is spontaneous emission for a single qubit instead of depolarizing noise
- **Ratio of Single-Qubit Spontaneous Emission in Two-Qubit Gate to  $p_2$  ( $p_2_{\text{emission\_ratio}}$ ):** fraction of  $p_2$  that is spontaneous emission for a single qubit in a two-qubit gate instead of depolarizing noise

The single and two-qubit fault probabilities are largely modeled using depolarizing channels; however, there is smaller probability that a spontaneous emission event happens. The probability is about an order of magnitude lower than the corresponding depolarizing error rate. The spontaneous emission error rates can be scaled using the scaling parameters given in the Scaling section. If a spontaneous emission event happens then  $\frac{1}{4}$  the time  $X$  is applied,  $\frac{1}{4}$  the time  $Y$  is applied, and  $\frac{1}{2}$  the time leakage is applied. For more details see: [Realization of Real-Time Fault-Tolerant Quantum Error Correction](#).

The two-qubit fault probability corresponds to the depolarizing probability of the System Model H2 fully entangling two-qubit gate,  $ZZ()$ . The probability of depolarizing error for the arbitrary angle two-qubit gate,  $RZZ(\theta)$ , depends on the angle  $\theta$ . The spontaneous emission error channel is the same for both  $ZZ()$  and  $RZZ(\theta)$ .

## Dephasing Noise

The noise model includes a memory error for which  $Z$  is applied. This is often called "dephasing" or "memory" noise. We potentially model two types of dephasing noise: one where the probability of applying  $Z$  is quadratically dependent on the duration for which qubits are idling or transporting in the trap, and another where the probability is linearly dependent on the duration. Note, we apply both sorts of noise simultaneously.

For state-vector simulations, the quadratic noise is modeled in the emulator by default as coherent noise. For this coherent quadratic dephasing noise, the  $RZ$  gate is applied with an angle proportional to frequency  $\times$  duration. The probability of the  $RZ$  gate applying a  $Z$  operation on a plus state is  $\sin(\text{frequency} \times \text{duration}/2)^2$ , which is why we call this a form of quadratic dephasing.

For the stabilizer simulator, by default this quadratic noise is modelled incoherent by applying Pauli  $Z$  with probability  $\sin(\text{frequency} \times \text{duration}/2)^2$  to model more closely the quadratic dependency with frequency and time as seen in the coherent model. Note, stabilizer simulations can only simulate Clifford and measurement-like gates, so the  $RZ$  gate cannot be applied directly. For both the state-vector and stabilizer simulations, linear dephasing is also modeled with  $Z$  applied with probability rate  $\times$  duration.

Switching between the coherent and incoherent quadratic dephasing model can be accomplished by setting `coherent_dephasing` either True or False. As mentioned, `coherent_dephasing` is True by default for the state-vector simulations and False by default for stabilizer simulations. If `coherent_dephasing` is set to False then the frequency for the quadratic error model (`quadratic_dephasing_rate`) is multiplied by `coherent_to_incoherent_factor` to attempt to make up for increased noise due to coherent effects; however, how sensitive circuits are to coherent effects depends on the circuit. Therefore, users may want to adjust this factor appropriately.

In addition, a transport dephasing parameter (`transport_dephasing`) is turned on by default and an idle dephasing parameter is turned on by default (`idle_dephasing`). Both of these can be toggled off.

- **Coherent Quadratic Dephasing Model** : the gate  $RZ$  (frequency x duration) is applied during transport and qubit idling, applied by default for the state-vector simulator, where frequency is equal to `quadratic_dephasing_rate` (in usings of  $2\pi$  rads per sec). This model is used if `coherent_dephasing` is True.
- **Incoherent Quadratic Dephasing Model**: Pauli  $Z$  is applied during transport and qubit idling according to the probability  $\sin(\text{frequency} \times \text{duration}/2)^2$ , applied by default for the stabilizer simulator, where frequency is equal to `quadratic_dephasing_rate` x `coherent_to_incoherent_factor` (all in units of  $2\pi$  radians per second). This model is used if `coherent_dephasing` is False. This model is mostly used to mimic coherent dephasing noise for stabilizer simulations.
- **Linear Dephasing Model**: Pauli  $Z$  is applied during transport and qubit idling according to the probability of `linear_dephasing_rate` x duration (where `linear_dephasing_rate` is in units of per sec. and duration is in units of seconds). This model is used in conjunction with both the coherent or incoherent quadratic dephasing model.

## Arbitrary Angle Noise Scaling

The System Model H2 systems have a native arbitrary-angle  $ZZ$  gate,  $RZZ(\theta)$ . For implementation of this gate in the System Model H2 Emulator, certain parameters relate to the strength of the depolarizing noise. These parameters depend on the angle  $\theta$ . This is normalized so that  $\theta = \frac{\pi}{2}$  gives the two-qubit fault probability ( $p2$ ).

The parameters for depolarizing noise are fit parameters that fit the noise estimated as the angle  $\theta$  changes per this equation:

$$\begin{aligned} & \left( przz_a \sqrt{|\theta|/\pi} + przz_b \right) * p2 && \text{for } \theta < 0 \\ & \left( przz_c \sqrt{|\theta|/\pi} + przz_d \right) * p2 && \text{for } \theta > 0 \\ & 0 && \text{for } \theta = 0 \end{aligned}$$

- Fit Parameter 1 (`przz_a`)
- Fit Parameter 2 (`przz_b`)
- Fit Parameter 3 (`przz_c`)
- Fit Parameter 4 (`przz_d`)



## Scaling

A scaling factor can be applied that multiplies all the default or supplied error parameters by the scaling rate. In this case, a 1 does not change the error rates while 0 makes all the errors have a probability of 0. Other aspects of the noise model can scale specific error rates in the error model, which include:

- **Scaling (scale):** scale all error rates in the model linearly
- **P1 Scaling (p1\_scale):** scale the probability of single-qubit gates having a fault
- **P2 Scaling (p2\_scale):** scale the probability of two-qubit gates having a fault
- **Measurement Scaling (meas\_scale):** scale the probability of measurement having a fault
- **Initialization Scaling (init\_scale):** scale the probability of initialization having a fault
- **Memory Scaling (memory\_scale):** linearly scale the probability of dephasing causing a fault
- **Emission Scaling (emission\_scale):** scale the probability that a spontaneous emission event happens during a single or two-qubit gate
- **Cross-talk Scaling (crosstalk\_scale):** scale the probability that measurement or initialization crosstalk events get applied to qubits, during mid-circuit measurement and reset (initialization), "crosstalk" noise can occur that effectively measures other qubits in the trap or cause them to leak.
- **Leakage Scaling (leakage\_scale):** scale the probability that a leakage even happens during single or two-qubit gates as well as during initialization or crosstalk; on the device half the time, spontaneous emission leads to a leakage event

## ■ APPENDIX

A H-System Quantum Credit (HQC) is defined as:

$$HQC = 5 + \frac{N_{1q} + 10 N_{2q} + 5 N_m}{5000} C$$

where  $N_{1q}$  is the number of single-qubit gates,  $N_{2q}$  is the number of native two-qubit gates,  $N_m$  is the number of state preparation and measurement operations in a circuit, including the initial implicit state preparation and any intermediate and final measurements and state resets, and  $C$  is the shot count. When a circuit is submitted (whether to the Syntax Checker, System Model H2, or System Model H2 Emulator) the cost in HQCs is returned with the results.

■