Task results

Module to represent the various enums and classes to assist with task results after execution on QuEra system.

QuEraShotResult

Bases: `BaseModel`

Object representing results after executing a task in QuEra system.

Attributes:

Name	Type	Description
`shot_status`	`QuEraShotStatusCode`	status code of task after running on QuEra system. Defaults to `QuEraShotStatusCode.MissingMeasurement`
`pre_sequence`	`conlist(conint(ge=0, le=1), min_items=0)`	Set of preprocessing instructions.
`post_sequence`	`conlist(conint(ge=0, le=1), min_items=0)`	Set of preprocessing instructions.

QuEraShotStatusCode

Bases: `str`, `Enum`

An Enum representing the status code the task executed on the QuEra system.

Attributes:

Name	Type	Description
`Completed`		The task has completed successfully.

QuEraTaskResults

Bases: `BaseModel`

Object representing results after executing a task in QuEra system.

Attributes:

Name	Type	Description
`task_status`	`QuEraShotStatusCode`	states of task in the QuEra system. Defaults to `QuEraShotStatusCode.Failed`
`shot_outputs`	`conlist(QuEraShotResult, min_items=0`	list representing shot outputs from QuEra system.

export_as_probabilities

```
export_as_probabilities()
```

converts from shot results to probabilities

Returns:

Name	Type	Description
`TaskProbabilities`	`TaskProbabilities`	The task results as probabilties

Source code in `src\bloqade\submission\ir\task_results.py`

```
151 def export_as_probabilities(self) -> TaskProbabilities:
152     """converts from shot results to probabilities
153 
154     Returns:
155         TaskProbabilities: The task results as probabilties
156     """
157     counts = dict()
158     nshots = len(self.shot_outputs)
159     for shot_result in self.shot_outputs:
160         pre_sequence_str = "".join(str(bit) for bit in
161 shot_result.pre_sequence)
162 
163         post_sequence_str = "".join(str(bit) for bit in
164 shot_result.post_sequence)
165 
166         configuration = (pre_sequence_str, post_sequence_str)
167         # iterative average
168         current_count = counts.get(configuration, 0)
169         counts[configuration] = current_count + 1
170 
    probabilities = [(config, count / nshots) for config, count in
counts.items()]
    return TaskProbabilities(probabilities=probabilities)
```

QuEraTaskStatusCode

Bases: `str`, `Enum`

An Enum representing the various states a task can be in within the QuEra system.

Attributes:

Name	Type	Description
`Created`		The task has been created but not yet started.
`Running`		The task is currently running.
`Completed`		The task has completed successfully.
`Failed`		The task has failed.

Name	Type	Description
`Cancelled`		The task has been cancelled.
`Executing`		The task is currently being executed.
`Enqueued`		The task is in the queue waiting to be executed.
`Accepted`		The task has been accepted for execution.
`Unaccepted`		The task has not been accepted for execution.
`Partial`		The task has partially completed.
`Unsubmitted`		The task has not been submitted for execution.

TaskProbabilities

Bases: `BaseModel`

The task results as probabilties.

Attributes:

Name	Type	Description
`probabilities`	`List[Tuple[Tuple[str, str], float]]`	task results as probabilities

simulate_task_results

```
simulate_task_results(shots=1)
```

Simulate the task results as probabilties.

Parameters:

Name	Type	Description	Default
`shots`	`int`	Number of shots, Defaults to 1.	`1`

Returns:

Name	Type	Description
`task_result`	`QuEraTaskResults`	Result of task simulation For example:

```
{
    "task_status": "Completed",
    "shot_outputs": [
        {
        "shot_status": "Completed",
        "pre_sequence": [1],
        "post_sequence": [1]
        },
        ....
        {
        "shot_status": "Completed",
        "pre_sequence": [1],
        "post_sequence": [1]
        }
    ],
}
```

Source code in `src\bloqade\submission\ir\task_results.py`

```python
def simulate_task_results(self, shots: int = 1) -> "QuEraTaskResults":
    """Simulate the task results as probabilties.

    Args:
        shots (int): Number of shots, Defaults to 1.

    Returns:
        task_result (QuEraTaskResults): Result of task simulation
            For example:
            ```python
            {
                "task_status": "Completed",
                "shot_outputs": [
                    {
                    "shot_status": "Completed",
                    "pre_sequence": [1],
                    "post_sequence": [1]
                    },
                    ....
                    {
                    "shot_status": "Completed",
                    "pre_sequence": [1],
                    "post_sequence": [1]
                    }
                ],
            }
            ```
    """
    bit_strings, probabilities = zip(*self.probabilities)

    indices = np.random.choice(len(probabilities), p=probabilities,
size=shots)
    shot_outputs = []
    for index in indices:
        pre_string, post_string = bit_strings[index]
        pre_sequence = [int(bit) for bit in pre_string]
        post_sequence = [int(bit) for bit in post_string]

        shot_outputs.append(
            QuEraShotResult(
                shot_status=QuEraShotStatusCode.Completed,
                pre_sequence=pre_sequence,
                post_sequence=post_sequence,
)
)

    return QuEraTaskResults(
        task_status=QuEraTaskStatusCode.Completed, shot_outputs=shot_outputs
)
```