Result_analysis

March 1, 2022

1 Analysis the results of hybrid PBC

In this notebook, we will analyse the results obtained for the **Toffoli toy example** using our hybrid PBC script.

We run the script simulating **1 virtual qubit** and demanding a precision $\epsilon_{desired} = 0.01$.

The number of samples is being determined as:

$$N = \frac{20 \cdot \sum_{i=1}^{\chi} \alpha_i^2}{\epsilon_{desired}^2},$$

and the actual relative error ϵ_{actual} as:

$$\epsilon_{actual} = \sqrt{20}\frac{\sigma_{samples}}{\sqrt{N}}.$$

Note that since necessarily $\sigma_{samples}^2 \leq \sum_{i=1}^{\chi} \alpha_i^2$ then $\epsilon_{actual} \leq \epsilon_{desired}$.

As a result, we will have 95%-confidence intervals given by $\mathbb{E}(\xi) \pm \epsilon_{actual}$.

[1]:
```
import math
import json

import numpy as np

from math import sqrt
from matplotlib import pyplot as plt
```

[2]:
```
def nr_samples(precision, virtual_qubits):
    weights_1vq = [1 / 2, (1 - sqrt(2)) / 2, 1 / sqrt(2)]

    sum_squares = 0
    for i in range(3**virtual_qubits):
        label = np.base_repr(i, base=3)
        label = str(label)
        if not len(label) == virtual_qubits:
            label = '0' * (virtual_qubits - len(label)) + label

        weight = 1
        for s in label:
```

```
            weight = weight * weights_1vq[int(s)]

        sum_squares += abs(weight)**2

    N = int(math.ceil(20 * sum_squares / precision**2))

    return N
```

[3]:
```
precision = [0.1, 0.09, 0.07, 0.05, 0.03, 0.02, 0.01]
vq = 1  # choosing 1 virtual qubit
samples = []

for p in precision:
    samples.append(nr_samples(p, vq))

print(samples)
```

```
[1586, 1958, 3237, 6344, 17620, 39645, 158579]
```

Because we run the code for the precision $\epsilon = 0.01$, we should have a total of 158 579 values to use for the computation of the probability p. We can use those values to estimate the expected value of this probability and determine how close it is to the exact value $p = 1$.

We can also determine the standard deviation of the sample, and use it to compute the appropriate confidence interval.

Since we have access to smaller sets of values as well, we can use the first M values to estimate the results that would have been obtained if we have requested a different precision (e.g., $\epsilon = \{0.10, 0.09, 0.07, 0.05, 0.03, 0.02\}$).

[4]:
```
exact_value = 1
```

1.1 $\epsilon = 0.01 \Rightarrow N = 158\,579$

[5]:
```
with open('Resources_data--virt1--probabilities.txt', 'r') as file_object:
    probabilities1 = json.load(file_object)
```

[6]:
```
probabilities1 = np.array(probabilities1)
prec = precision[-1]
print(len(probabilities1) == samples[-1])

mean1 = np.mean(probabilities1)
std1 = np.std(probabilities1)
error1 = sqrt(20) * std1 / sqrt(len(probabilities1))

print(f'p= {round(mean1, 4)} +/- {round(error1, 4)}')
print(error1 < prec)
```

```
True
p= 0.9997 +/- 0.0035
True
```

1.2 $\epsilon = 0.02 \Rightarrow N = 39\,645$

[7]:
```python
probabilities2 = probabilities1[:samples[-2]]
```

[8]:
```python
prec = precision[-2]
print(len(probabilities2) == samples[-2])

mean2 = np.mean(probabilities2)
std2 = np.std(probabilities2)
error2 = sqrt(20) * std2 / sqrt(len(probabilities2))

print(f'p= {round(mean2, 4)} +/- {round(error2, 4)}')
print(error2 < prec)
```

```
True
p= 0.9984 +/- 0.0071
True
```

1.3 $\epsilon = 0.03 \Rightarrow N = 17\,620$

[9]:
```python
probabilities3 = probabilities1[:samples[-3]]
```

[10]:
```python
prec = precision[-3]
print(len(probabilities3) == samples[-3])

mean3 = np.mean(probabilities3)
std3 = np.std(probabilities3)
error3 = sqrt(20) * std3 / sqrt(len(probabilities3))

print(f'p= {round(mean3, 4)} +/- {round(error3, 4)}')
print(error3 < prec)
```

```
True
p= 0.9979 +/- 0.0107
True
```

1.4 $\epsilon = 0.05 \Rightarrow N = 6\,344$

[11]:
```python
probabilities4 = probabilities1[:samples[-4]]
```

[12]:
```python
prec = precision[-4]
print(len(probabilities4) == samples[-4])

mean4 = np.mean(probabilities4)
```

```python
std4 = np.std(probabilities4)
error4 = sqrt(20) * std4 / sqrt(len(probabilities4))

print(f'p= {round(mean4, 4)} +/- {round(error4, 4)}')
print(error4 < prec)
```

```
True
p= 0.9972 +/- 0.0178
True
```

1.5 $\epsilon = 0.07 \Rightarrow N = 3\,237$

[13]:
```python
probabilities5 = probabilities1[:samples[-5]]
```

[14]:
```python
prec = precision[-5]
print(len(probabilities5) == samples[-5])

mean5 = np.mean(probabilities5)
std5 = np.std(probabilities5)
error5 = sqrt(20) * std5 / sqrt(len(probabilities5))

print(f'p= {round(mean5, 4)} +/- {round(error5, 4)}')
print(error5 < prec)
```

```
True
p= 0.9961 +/- 0.0248
True
```

1.6 $\epsilon = 0.09 \Rightarrow N = 1\,958$

[15]:
```python
probabilities6 = probabilities1[:samples[-6]]
```

[16]:
```python
prec = precision[-6]
print(len(probabilities6) == samples[-6])

mean6 = np.mean(probabilities6)
std6 = np.std(probabilities6)
error6 = sqrt(20) * std6 / sqrt(len(probabilities6))

print(f'p= {round(mean6, 4)} +/- {round(error6, 4)}')
print(error6 < prec)
```

```
True
p= 0.9878 +/- 0.0323
True
```

1.7 $\epsilon = 0.100 \Rightarrow N = 1\,586$

[17]:
```
probabilities7 = probabilities1[:samples[-7]]
```

[18]:
```
prec = precision[-7]
print(len(probabilities7) == samples[-7])

mean7 = np.mean(probabilities7)
std7 = np.std(probabilities7)
error7 = sqrt(20) * std7 / sqrt(len(probabilities7))

print(f'p= {round(mean7, 4)} +/- {round(error7, 4)}')
print(error7 < prec)
```

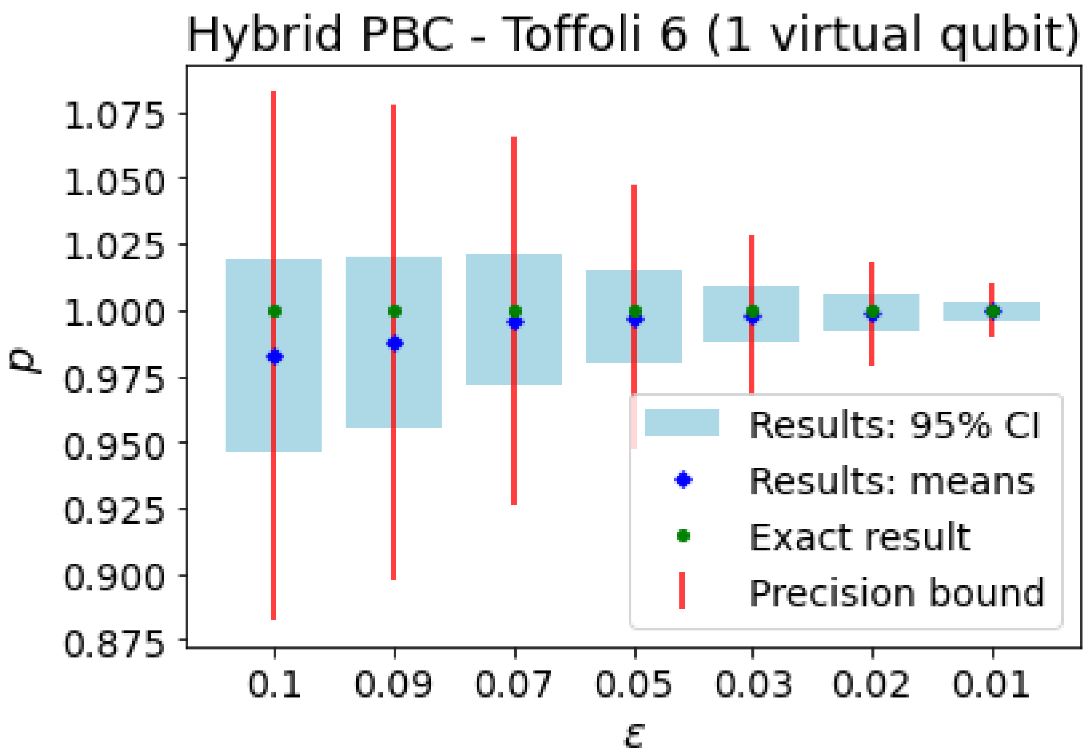
```
True
p= 0.9826 +/- 0.0364
True
```

1.8 Plotting the data:

[19]:
```
means = [mean7, mean6, mean5, mean4, mean3, mean2, mean1]
errors = [error7, error6, error5, error4, error3, error2, error1]

bot_bar = [means[i] - errors[i] for i in range(len(means))]
heights = [2*errors[i] for i in range(len(means))]
```

[20]:
```
fig = plt.figure()

plt.bar([i for i in range(len(means))],
        heights,
        width=0.8,
        bottom=bot_bar,
        align='center',
        color='lightblue',
        label='Results: 95% CI')

plt.errorbar([i for i in range(len(means))],
             means,
             color='blue',
             marker='D',
             markersize=4,
             linestyle='None',
             label='Results: means')

plt.errorbar([i for i in range(len(means))],
             [exact_value for _ in range(len(means))],
```

```python
             color='green',
             marker='o',
             markersize=4,
             linestyle='None',
             label='Exact result')

plt.errorbar([i for i in range(len(means))],
             means,
             yerr=precision,
             color='red',
             marker='None',
             markersize=4,
             linestyle='None',
             label='Precision bound')

plt.xticks([i for i in range(len(means))], precision, size=14)
plt.xlabel(r'$\epsilon$', fontsize=16)

#plt.ylim([0.05, 0.25])
plt.yticks(size=14)
plt.ylabel(r'$p$', fontsize=16)

plt.legend(fontsize=14)

plt.title('Hybrid PBC - Toffoli 6 (1 virtual qubit)', fontsize=18)

plt.show()
```



Note that since p cannot be larger than 1, it is clear that we can stop the confidence intervals (and the precision bounds) at that value:

[21]:
```
trimmed_heights = []
for i in range(len(heights)):
    trimmed_heights.append(1 - bot_bar[i])

trimmed_max_err = []
for i in range(len(means)):
    trimmed_max_err.append(1 - means[i])
```

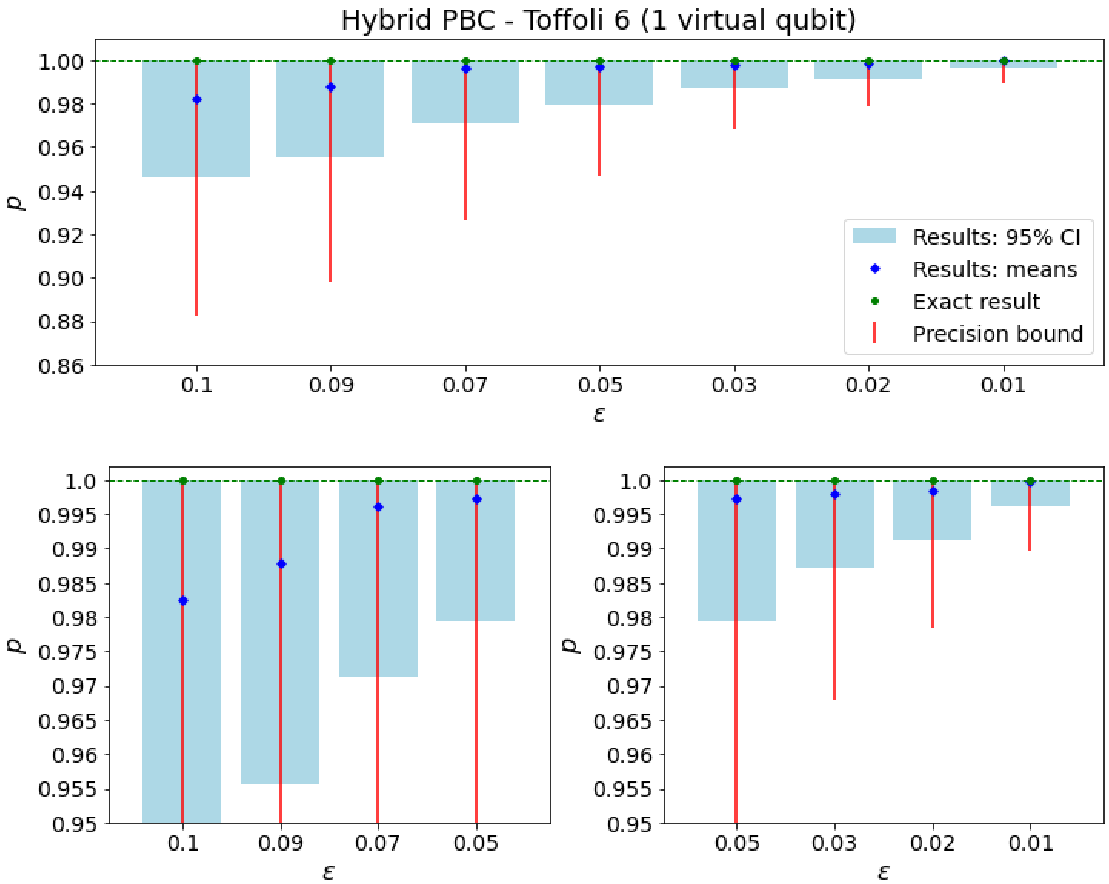
[22]:
```
fig = plt.figure(constrained_layout=True, figsize=(10, 8))

subfigs = fig.subfigures(2, 1, hspace=0.07)

# Upper subfigure
subplot = subfigs[0].add_subplot(111)
plt.bar([i for i in range(len(means))],
        trimmed_heights,
        width=0.8,
        bottom=bot_bar,
        align='center',
        color='lightblue',
```

```
        label='Results: 95% CI')

plt.errorbar([i for i in range(len(means))],
             means,
             color='blue',
             marker='D',
             markersize=4,
             linestyle='None',
             label='Results: means')

plt.errorbar([i for i in range(len(means))],
             [exact_value for _ in range(len(means))],
             color='green',
             marker='o',
             markersize=4,
             linestyle='None',
             linewidth=1,
             label='Exact result')

plt.errorbar([i for i in range(len(means))],
             means,
             yerr=[precision, trimmed_max_err],
             color='red',
             marker='None',
             markersize=4,
             linestyle='None',
             label='Precision bound')

plt.plot(np.arange(-1, 8, 1), [exact_value for _ in np.arange(-1, 8, 1)],
         '--',
         color='green',
         linewidth=1)

plt.xlim([-0.75, 6.75])
plt.xticks([i for i in range(len(means))], precision, size=14)
plt.xlabel(r'$\epsilon$', fontsize=16)

plt.ylim([0.86, 1.01])
plt.yticks(size=14)
plt.ylabel(r'$p$', fontsize=16)

plt.legend(fontsize=14, loc=4)

plt.title('Hybrid PBC - Toffoli 6 (1 virtual qubit)', fontsize=18)

# Lower subfigure (left)
subplot = subfigs[1].add_subplot(121)
```

```
plt.bar([i for i in range(4)],
        trimmed_heights[:4],
        width=0.8,
        bottom=bot_bar[:4],
        align='center',
        color='lightblue',
        label='Results: 95% CI')

plt.errorbar([i for i in range(4)],
             means[:4],
             color='blue',
             marker='D',
             markersize=4,
             linestyle='None',
             label='Results: means')

plt.errorbar([i for i in range(4)], [exact_value for _ in range(4)],
             color='green',
             marker='o',
             markersize=4,
             linestyle='None',
             linewidth=1,
             label='Exact result')

plt.errorbar([i for i in range(4)],
             means[:4],
             yerr=[precision[:4], trimmed_max_err[:4]],
             color='red',
             marker='None',
             markersize=4,
             linestyle='None',
             label='Precision bound')

plt.plot(np.arange(-1, 8, 1), [exact_value for _ in np.arange(-1, 8, 1)],
         '--',
         color='green',
         linewidth=1)

plt.xlim([-0.75, 3.75])
plt.xticks([i for i in range(4)], precision[:4], size=14)
plt.xlabel(r'$\epsilon$', fontsize=16)

plt.ylim([0.95, 1.002])
plt.yticks(np.arange(0.95, 1.002, 0.005),
           [round(i, 3) for i in np.arange(0.95, 1.002, 0.005)],
           size=14)
plt.ylabel(r'$p$', fontsize=16)
```

```
# Lower subfigure (right)
subplot = subfigs[1].add_subplot(122)
plt.bar([i for i in range(4)],
        trimmed_heights[3:],
        width=0.8,
        bottom=bot_bar[3:],
        align='center',
        color='lightblue',
        label='Results: 95% CI')

plt.errorbar([i for i in range(4)],
             means[3:],
             color='blue',
             marker='D',
             markersize=4,
             linestyle='None',
             label='Results: means')

plt.errorbar([i for i in range(4)], [exact_value for _ in range(4)],
             color='green',
             marker='o',
             markersize=4,
             linestyle='None',
             linewidth=1,
             label='Exact result')

plt.errorbar([i for i in range(4)],
             means[3:],
             yerr=[precision[3:], trimmed_max_err[3:]],
             color='red',
             marker='None',
             markersize=4,
             linestyle='None',
             label='Precision bound')

plt.plot(np.arange(-1, 8, 1), [exact_value for _ in np.arange(-1, 8, 1)],
         '--',
         color='green',
         linewidth=1)

plt.xlim([-0.75, 3.75])
plt.xticks([i for i in range(4)], precision[3:], size=14)
plt.xlabel(r'$\epsilon$', fontsize=16)

plt.ylim([0.95, 1.002])
plt.yticks(np.arange(0.95, 1.002, 0.005),
```

```
            [round(i, 3) for i in np.arange(0.95, 1.002, 0.005)],
            size=14)
plt.ylabel(r'$p$', fontsize=16)

fig.savefig("Probability_vs_precision.pdf", bbox_inches='tight')
plt.show()
```



[ ]: