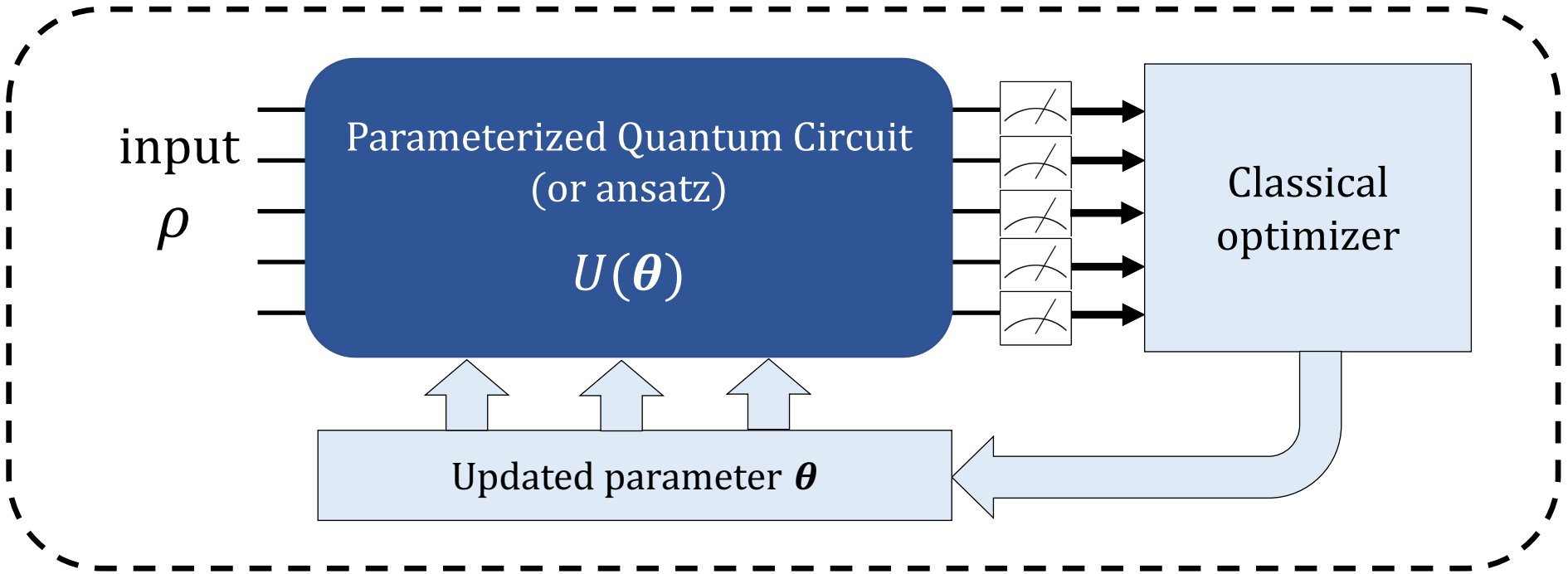
Suffix averaging technique for improving optimization result in VQAs

Shiro Tamiya (IBM Quantum, IBM Research — Tokyo Intern)

Mentor: Ikko Hamamura

Aug 18, 2022 Qiskit DemoDays

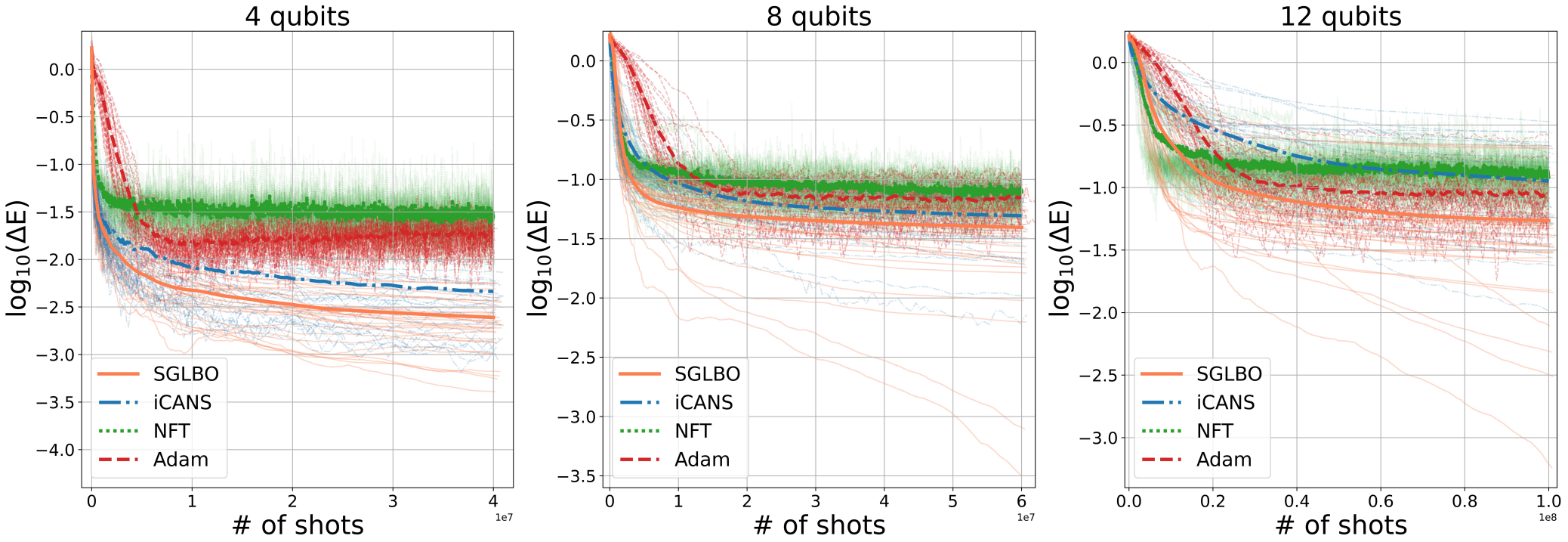
Optimization in VQAs



Task: $\boldsymbol{\theta}^* = \underset{\theta}{\operatorname{argmin}} f(\boldsymbol{\theta})$

Stochastic gradient line Bayesian optimization (SGLBO)

S. Tamiya and H. Yamasaki, npj Quant. Info. **8**:90 (2022)



Results

- Propose SGLBO by combining SGD for estimating gradients and BO for estimating optimal step size.
- Introduce two noise-reduction techniques, adaptive shot strategy and **suffix averaging**.
- SGLBO exhibits high accuracy and high noise resilience compared to state-of-the-art optimizers.

Suffix averaging (SA)

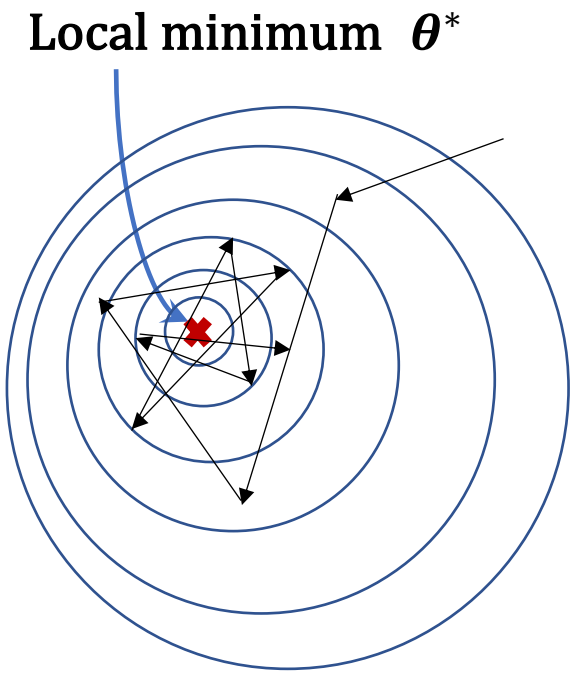
- In VQAs, we usually utilize the point $\widehat{\boldsymbol{\theta}}^{(T)}$ obtained by final iterate as a result of an optimization .
- However, the points $\{\widehat{\boldsymbol{\theta}}^{(t)}\}_t$ may oscillate around a local minimum due to statistical effects.

Iteration	1	2	⋯	⋯	$T-2$	$T-1$	T
Point	$\widehat{\boldsymbol{\theta}}^{(1)}$	$\widehat{\boldsymbol{\theta}}^{(2)}$	⋯	⋯	$\widehat{\boldsymbol{\theta}}^{(T-2)}$	$\widehat{\boldsymbol{\theta}}^{(T-1)}$	$\widehat{\boldsymbol{\theta}}^{(T)}$

N_{SA} points

$$\widehat{\boldsymbol{\theta}}^{*} \leftarrow \frac{1}{N_{\mathrm{SA}}} \sum_{i=1}^{N_{\mathrm{SA}}} \widehat{\boldsymbol{\theta}}^{(T-i+1)}$$

take average



Suffix averaging (SA)

```
optimizer = ADAM(maxiter=1000)
ansatz = TwoLocal(2, ['ry'], 'cx', "linear")
grad = Gradient(grad_method="param_shift")
```

[5] ✓ 0.1s Python

```
suffix_optimizer = SuffixAveragingOptimizer(optimizer, n_params_suffix=50)
```

[6] ✓ 0.2s Python

```
counts = []
values = []
def store_intermediate_result(eval_count, parameters, mean, std):
    counts.append(eval_count)
    values.append(mean)
vqe = VQE(ansatz, suffix_optimizer, callback = store_intermediate_result, gradient = grad, quantum_instance=quantum_instance)
```

[7] ✓ 0.2s Python

```
result = vqe.compute_minimum_eigenvalue(operator=h2_op)
```

[6] Python

```
print("Final point:", values[-2])
print("Suffix averaged point:", result.optimal_value)
print("Improvement:", values[-2]-result.optimal_value)
print("Optimal:", -1.85727503)
```

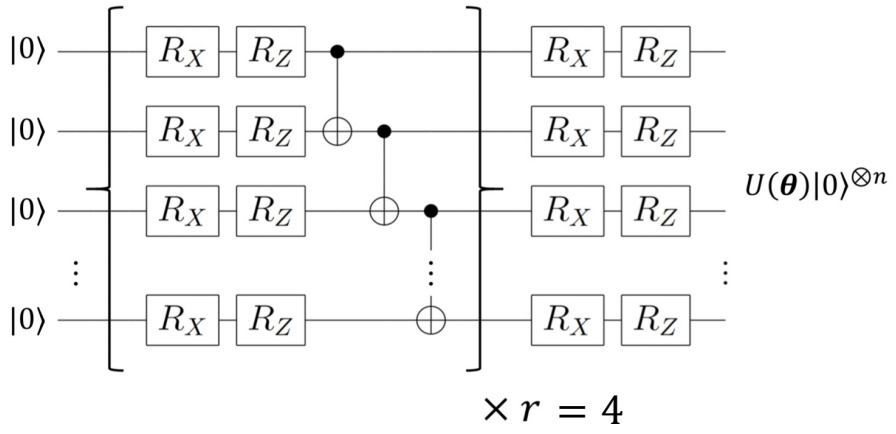
[18] Python

```
Final point: -1.8457205408261639
Suffix averaged point: -1.8522282693631102
Improvement: 0.006507728536946367
Optimal: -1.85727503
```

Numerical experiments: setups

- Model
  - H2 molecule
  - Transversal-field Ising model (TFIM) with open boundary condition

$$H = -J\sum_{i=1}^{n-1} Z_i Z_{i+1} - g\sum_{i=1}^{n} X_i \qquad \text{where } n = 4, 6$$

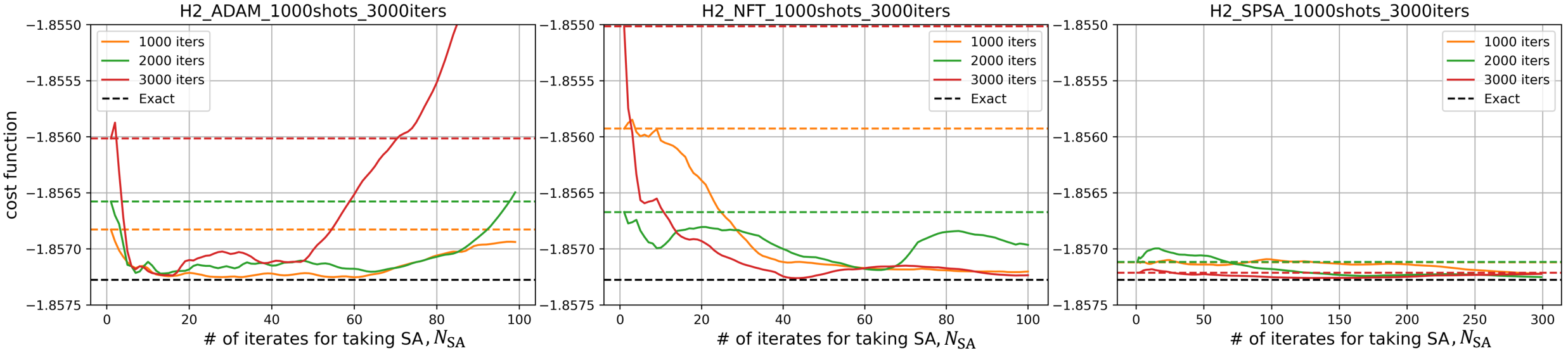


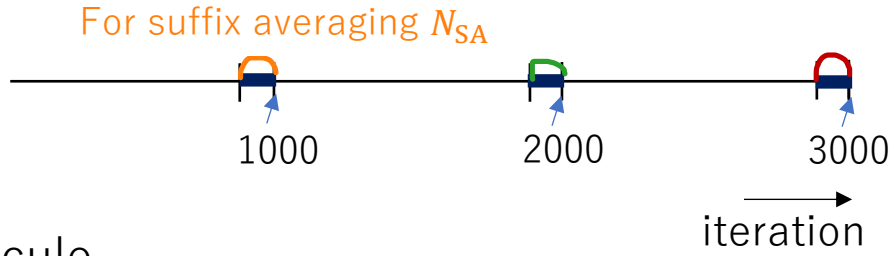
- Two local ansatz
- ADAM, Nakanishi-Todo-Fujii (NFT) method, SPSA
- Use Noiseless simulator

$$\partial_i f(\boldsymbol{\theta}) \sim \overset{\text{1000 shots}}{f\left(\boldsymbol{\theta} + \frac{\pi}{4}\boldsymbol{e}_i\right)} - \overset{\text{1000 shots}}{f\left(\boldsymbol{\theta} - \frac{\pi}{4}\boldsymbol{e}_i\right)}$$

- Consume $\mathbf{1000}$ shots for estimating each expectation value, and run 3000 iterations in total.

Numerical experiments 1: H2 molecule

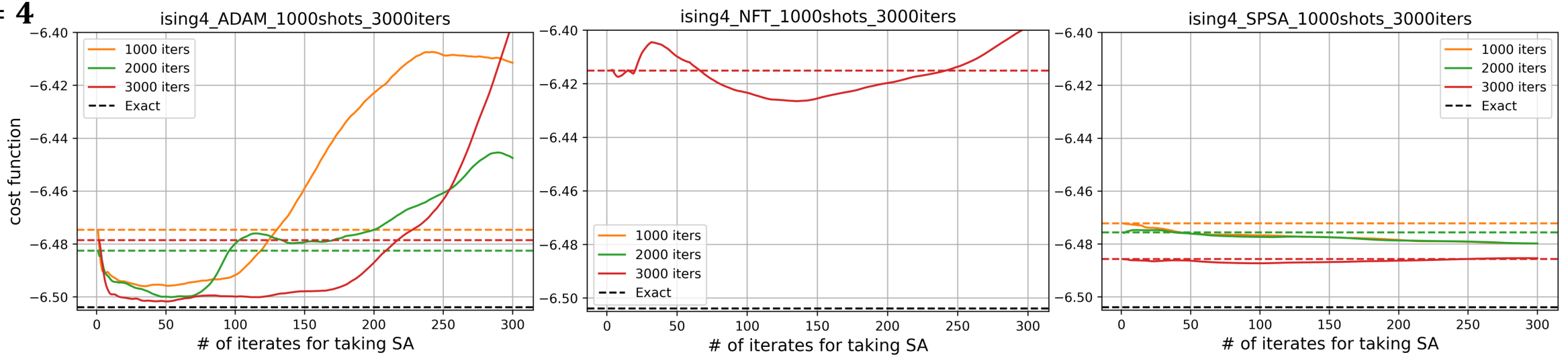




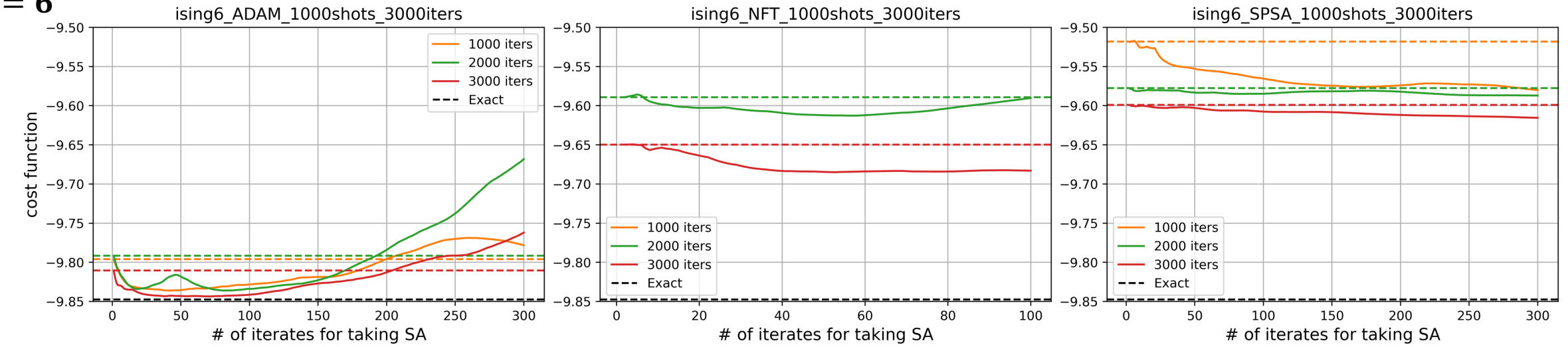
- Suffix averaging improves the final output in VQE of H2 molecule.
- Significant improvements are achieved in the case of ADAM and NFT.

Numerical experiments 2: TFIM with $n = 4, 6$

$n = 4$

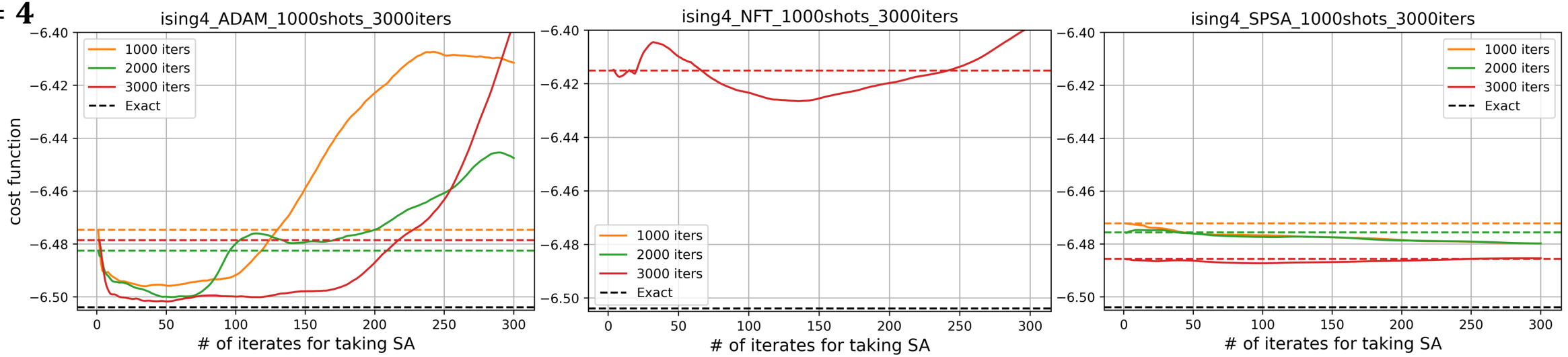


$n = 6$

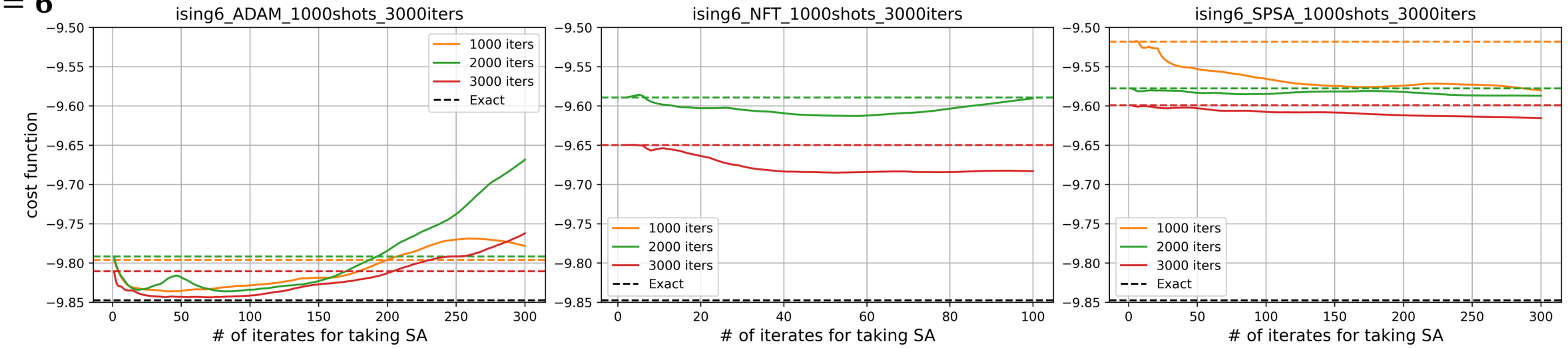


Numerical experiments 2: TFIM with $n = 4, 6$

$n = 4$

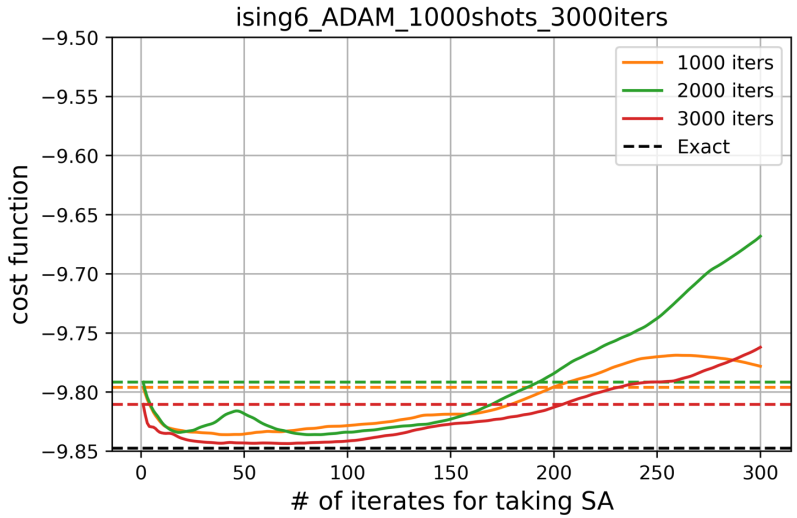


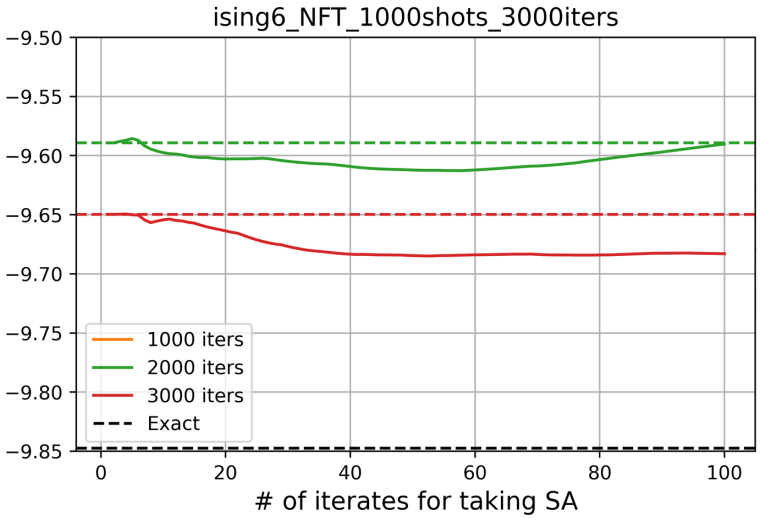
$n = 6$

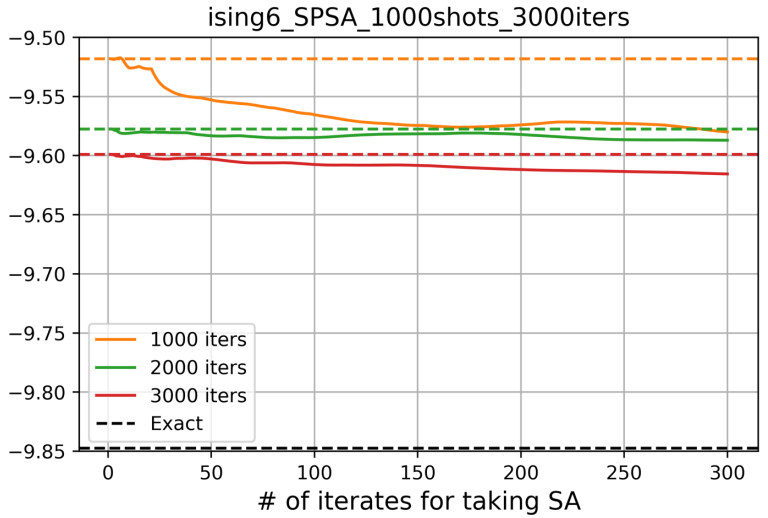


Numerical experiments 2: TFIM with $n = 6$

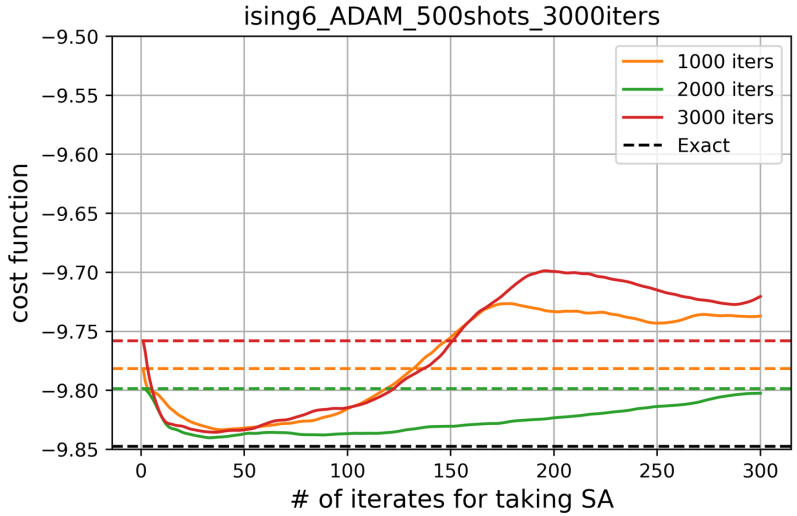
1000 shots

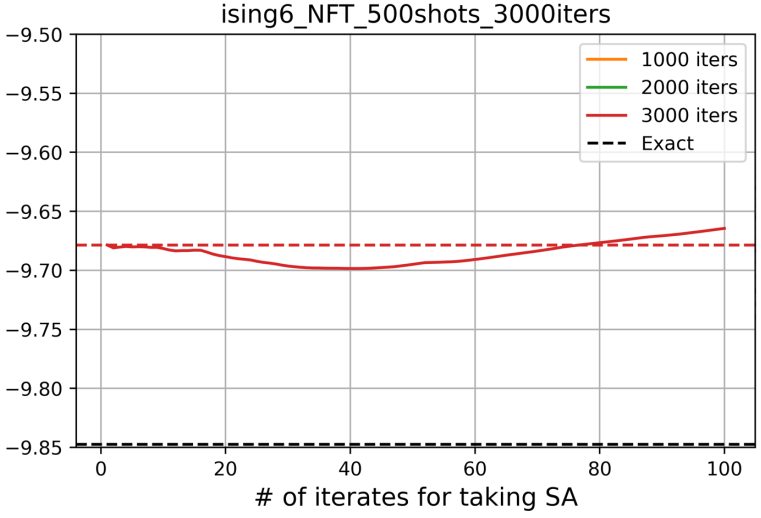


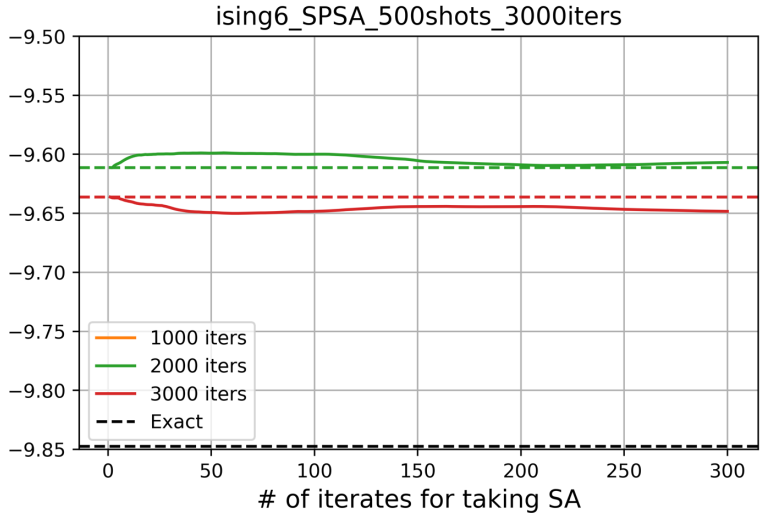




500 shots





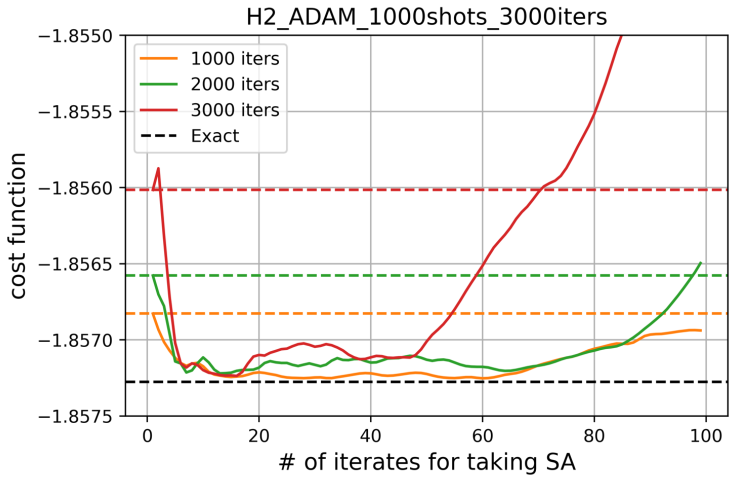


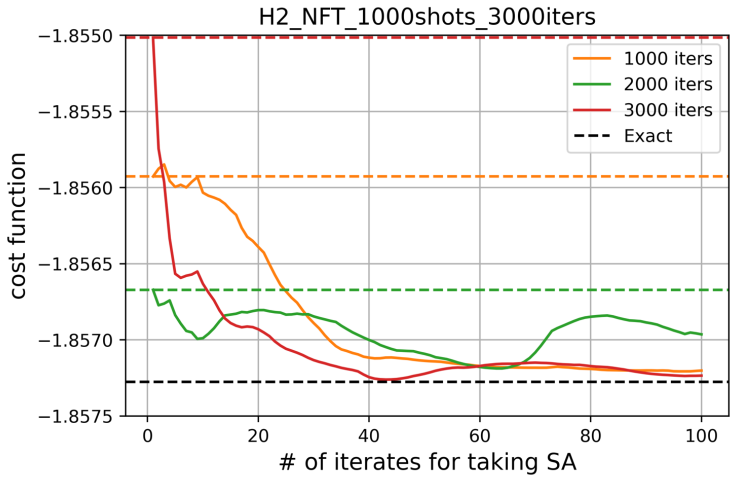
Conclusion

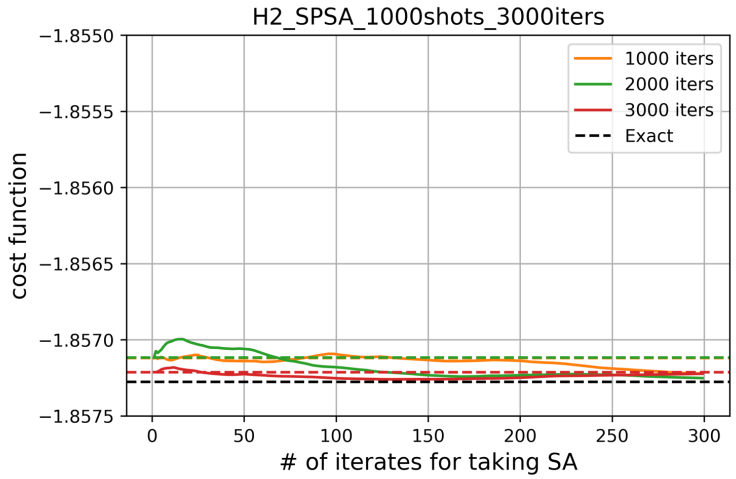
- By taking suffix averaging, we see the improvements of the results of the optimizations in almost every case.
- The advantage of ADAM+suffix averaging is remarkable when the system size increases.
- It is sensitive to the number of points N_{SA} used for suffix averaging.

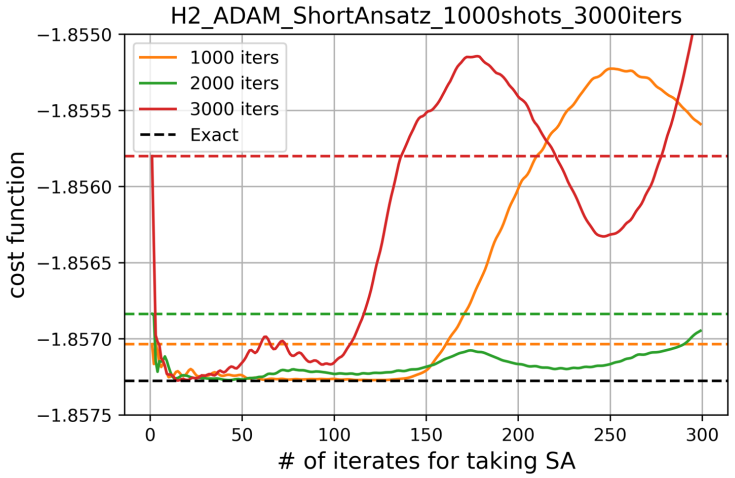
  $N_{\mathrm{SA}} = 50$ points may be preferrable.

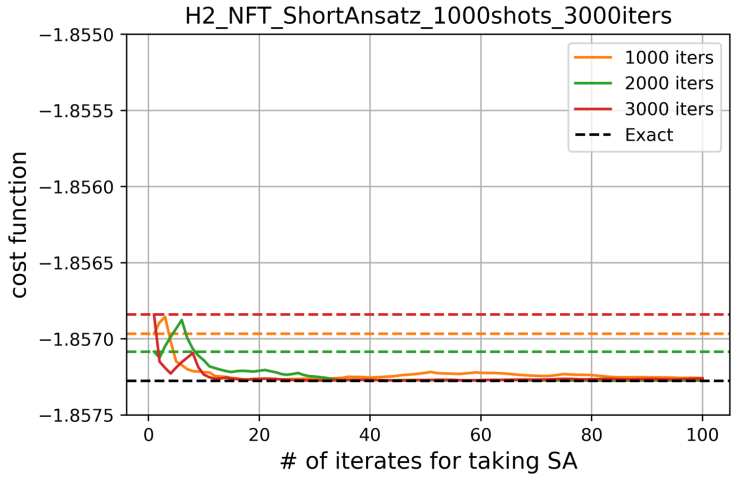
Numerical results 1: H2 molecule

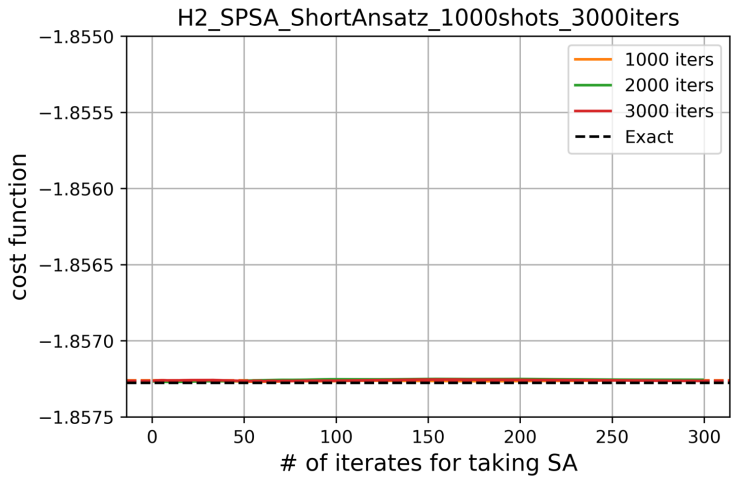




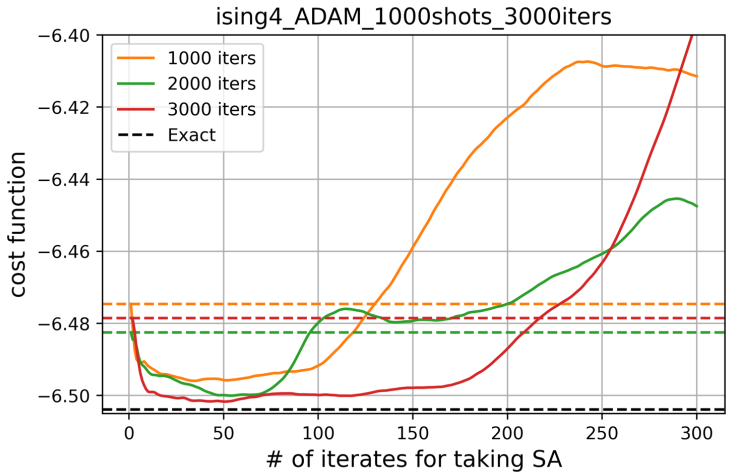


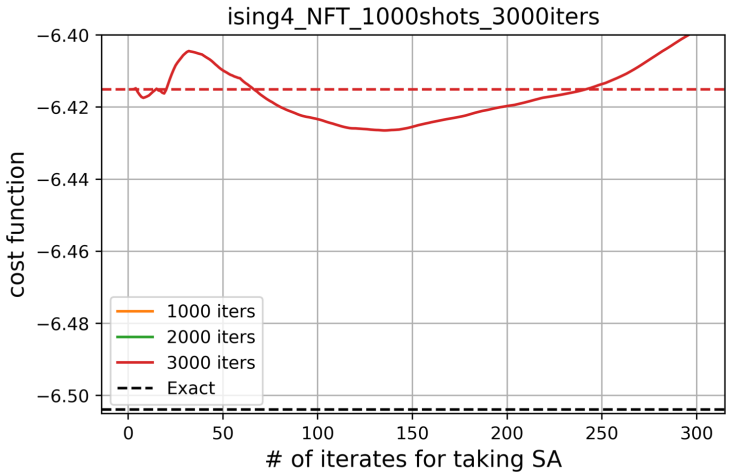


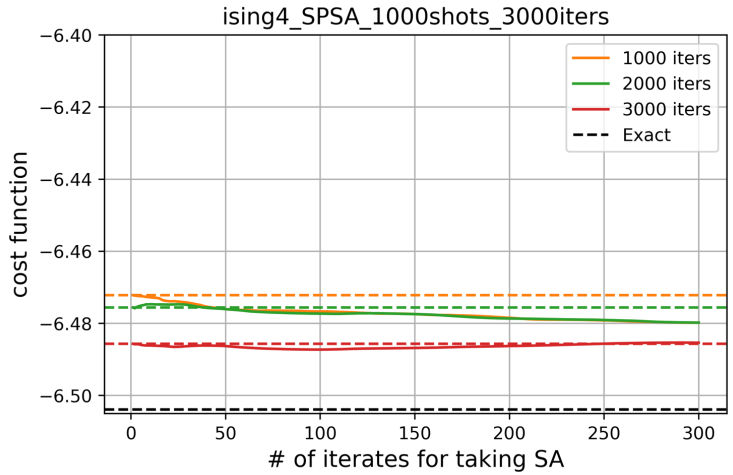


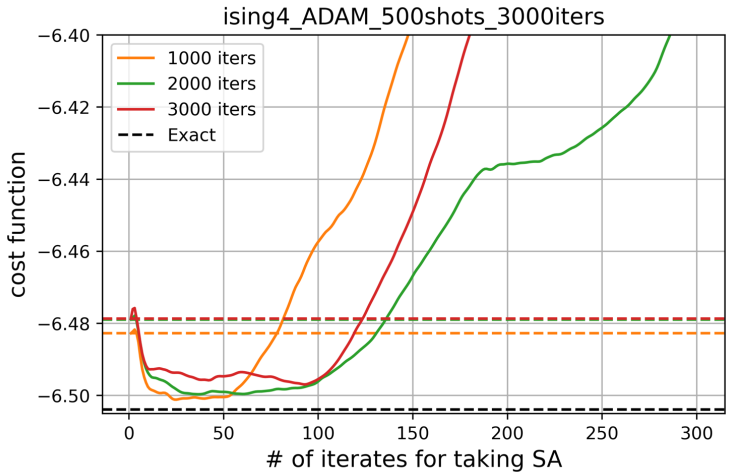


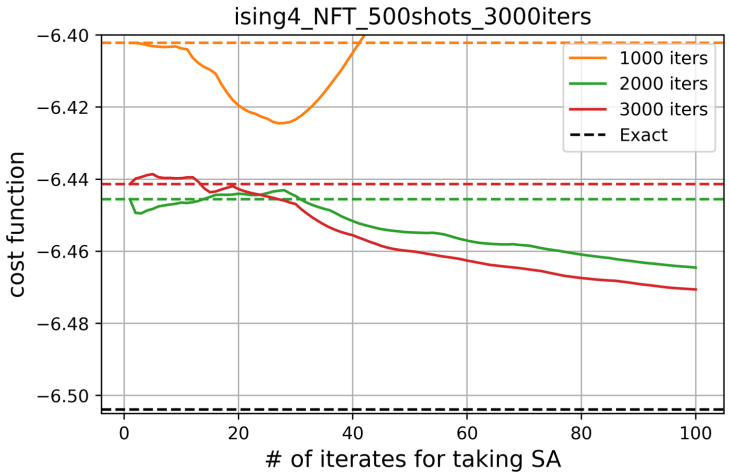
Numerical results 2: TFIM

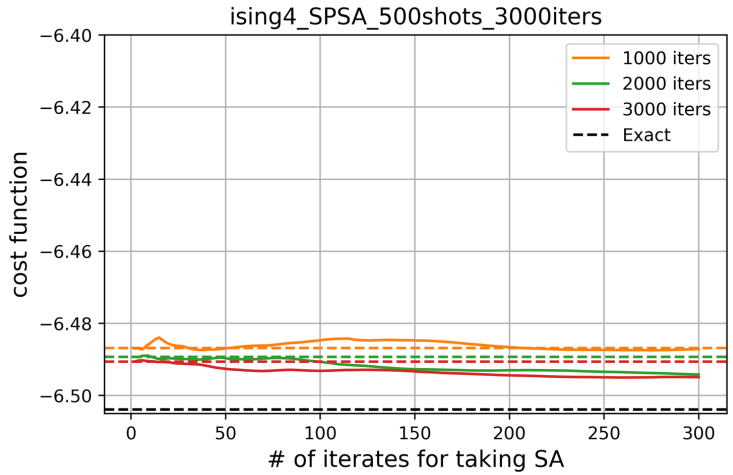












Numerical results 3: TFIM