Richardson Extrapolation

Quantum Wednesday

Nate Stemen

Feb 8, 2023

Today's Paper

Optimization of Richardson extrapolation for quantum error mitigation

Michael Krebsbach,[1,*] Björn Trauzettel,[1,2] and Alessio Calzona[1,2]

[1] *Institute for Theoretical Physics and Astrophysics, University of Würzburg, D-97074 Würzburg, Germany*
[2] *Würzburg-Dresden Cluster of Excellence ct.qmat, Germany*

(Dated: August 26, 2022)

Quantum error mitigation is a key concept for the development of practical applications based on current noisy intermediate scale quantum (NISQ) devices. One of the most promising methods is Richardson extrapolation to the zero noise limit. While its main idea is rather simple, the full potential of Richardson extrapolation has not been completely uncovered yet. We give an in-depth analysis of the relevant parameters of Richardson extrapolation and propose an optimized protocol for its implementation. This protocol allows for a precise control of the increase in statistical uncertainty and lays the foundation for a significant improvement of the mitigation performance achieved by increasing the number of nodes. Furthermore, we present a novel set of nodes that, on average, outperforms the linear, exponential or Chebyshev nodes frequently used for Richardson extrapolation without requiring any additional resources.

https://arxiv.org/abs/2201.08080

Overview

1. What is Richardson Extrapolation?
2. How do we *already* use it?
3. What sort of optimizations do the authors find?
4. Should we implement this?

What is extrapolation?

How do we use Richardson Extrapolation?

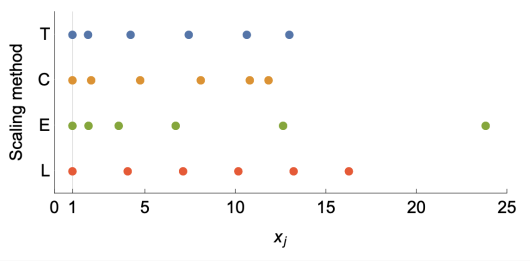
```
class RichardsonFactory():
  def extrapolate(
    scale_factors: Sequence[float],
    exp_values: Sequence[float],
    full_output: bool = False,
  ) -> ExtrapolationResult:
    # Richardson extrapolation is a particular case of a
    # polynomial fit with order equal to the number of
    # data points minus 1.
    order = len(scale_factors) - 1
    return PolyFactory.extrapolate(
      scale_factors, exp_values, order, full_output
    )
```

What could we optimize?

1. Is there a way to reduce the increase in variance?
2. Is there a scaling method for the x_i's that optimizes the bias/variance tradeoff?
3. Is there an optimal number of samples n?

Node Spacing

$$
\begin{aligned}
x_j^{\mathrm{L}} &= 1 + j(x_1 - 1) \\
x_j^{\mathrm{E}} &= x_1^j \\
x_j^{\mathrm{C}} &= 1 + \frac{\sin^2(\frac{j}{n}\frac{\pi}{2})}{\sin^2(\frac{1}{n}\frac{\pi}{2})}(x_1 - 1) \\
x_j^{\mathrm{T}} &= 1 + \frac{\sin^2(\frac{j}{n+1}\frac{\pi}{2})}{\sin^2(\frac{1}{n+1}\frac{\pi}{2})}(x_1 - 1)
\end{aligned}
$$



Thank you!