# Reverse-mode differentiation for quantum gradients 

Review of Efficient calculation of gradients in classical simulations of variational quantum algorithms
https://arxiv.org/abs/2009.02823

Motivation

- Optimizing variatival quentum circuits is everyhhere - URE, QAOA, etc.

Motivation

- Optimizing variational quantum circuits is evoryunere
- UCE, QAOA, etc.
- Goal: Minimize $[\omega L O G]$ the "energy"

$$
\left.E(\epsilon):=\langle\theta| H|\epsilon\rangle=\langle\operatorname{in}| U^{t}(\epsilon) H U(\epsilon) \mid \text { in }\right\rangle
$$

Motivation

- Optimizing variational quantum circuits is evoryutere
- U CE, QAOA, etc.
- Goal: Minimize $[\omega L O G]$ the "energy"

$$
E(\theta):=\langle\Theta| H|\theta\rangle=\langle i n| U^{t}(\theta) H U(e)|i n\rangle
$$



$$
u_{i}=u_{i}\left(c_{i}\right)
$$

$$
i=1 \ldots, p
$$

Motivation

- Optimizing variatival quantum circuits is evoryutere
- VEE, QNOA, etc.
- Goal: Minimize [ $\omega L 06]$ the "energy"

$\rightarrow$ Do this via gradient descent: $\Delta \theta \propto-\nabla E(c)$

Simple algorithm for the gradient
Assume I gates wi one unique parameter per gate.

Simple algorithm for the gradient
Assume I gates wi one unique parameter per gate.
for $i=1, \ldots, p$ :
Compute $\frac{\partial E}{\partial G_{i}}$

Simple algorithm for the gradient
Assume I gates wi one unique parameter per gate. for $i=1, \ldots, p:$

Compute $\frac{\partial E}{\partial G_{i}}$
$\rightarrow$ For eccuppl, vice finite difference

$$
\frac{\partial E}{\partial \theta_{i}} \approx \frac{E\left(\theta_{i}+\varepsilon\right)-E\left(\theta_{i}\right)}{\varepsilon}
$$

Simple algorithm for the gradient
Assume I gates wi one unique parameter per gate.
for $i=1, \ldots, p$ :
Compute $\frac{\partial E}{\partial G_{i}}$
$\rightarrow$ For eccuppl, vice finite difference

$$
\frac{\partial E}{\partial \theta_{i}} \approx \frac{E\left(\theta_{i}+\varepsilon\right)-E\left(\epsilon_{i}\right)}{\varepsilon}
$$

Complexity:

Simple algorithm for the gradient
Assume I gates wi one unique parameter per gate.
for $i=1, \ldots, p$ :
compute $\frac{\partial E}{\partial G_{i}}$
$\rightarrow$ For ecoupte, vise finite difference

$$
\frac{\partial E}{\partial \theta_{i}} \approx \frac{E\left(\theta_{i}+\varepsilon\right)-E\left(\epsilon_{i}\right)}{\varepsilon}
$$

Complexity:

- To compote $\frac{\partial E}{\partial G_{i}}$, we simulate $O(P)$ gates.

Simple algorithm for the gradient
Assume I gates wi one unique parameter per gate.
for $i=1, \ldots, p$ :
Compute $\frac{\partial E}{\partial G_{i}}$

$$
\frac{\partial E}{\partial \theta_{i}} \approx \frac{E\left(\theta_{i}+\varepsilon\right)-E\left(\sigma_{i}\right)}{\varepsilon}
$$

Complexity:

- To compote $\frac{\partial E}{\partial G_{i}}$ we simulate $O(P)$ gates.
- There are P partial derivatives to compote.

Simple algorithm for the gradient
Assume I gates wi one unique parameter per gate.
for $i=1, \ldots, p$ :
Compute $\frac{\partial E}{\partial G_{i}}$

For example, wise finite difference

$$
\frac{\partial E}{\partial \theta_{i}} \approx \frac{E\left(\theta_{i}+\varepsilon\right)-E\left(c_{i}\right)}{\varepsilon}
$$

Complexity:

- To compote $\frac{\partial E}{\partial \sigma_{i}}$ we simulate $O(P)$ gates.
- There are P partial derivatives to compute.

$$
\Rightarrow O\left(p^{2}\right) \text { complexity }
$$

Simple algorithm e for the gradient
Assume I gates wi one unique parameter per gate.

$$
\text { for } i=1, \ldots, p:
$$

compute $\frac{\partial E}{\partial G_{i}}$

$$
\frac{\partial E}{\partial \theta_{i}} \approx \frac{E\left(\theta_{i}+\varepsilon\right)-E\left(\sigma_{i}\right)}{\varepsilon}
$$

Complexity:

- To compote $\frac{\partial E}{\partial G_{i}}$, we simulate $O(P)$ gates.
- There are P partial derivatives to compute.

$$
\Rightarrow O\left(p^{2}\right) \text { complexity }
$$

* Question: Can we do better than $O\left(p^{2}\right)$ ?

Main Result

* (Description of) an $O(P)$ algorithm for the gradient $D E \ldots$

Main Result

* (Description of) an $O(P$ ) algorithm for the gradient DE...
...Under these cossumptions:
(1) State vector simulator
- ie, we have access to the full Wavefunction.

Main Result

* (Description of) an O(P) algorithm for the gradient DE...
... Under these cossumptions:
(1) State vector simulator
- ie, we have access to the foll wavefunction.
(2) Ability to apply the observille $H$ to a state $|\psi\rangle$
- ie, we can compute HIS

Main Result

* (Description of) an O(P) algorithm for the gradient DE...
...Under these cossumptions:
(1) State vector simulator
- ie, we have access to the foll wavefunction.
(2) Ability to apply the observille $H$ to a state $|\psi\rangle$
- ie, we can compute HIS
(3) Ability to compute gate derivatives and apply then to states
- ie, $\frac{d v_{i}}{d \theta_{i}}$ and $\frac{d v_{i}}{d \theta_{i}}|\psi\rangle$

Understanding the celgorithm
Input: Fixed input state $\operatorname{lin}\rangle$, unitary $U$, observable $H$.

Understanding the celgorithm
Input: Fixed input state line, unity $U$, observable $H$. the eth compenert of the gradient is

$$
\frac{\partial E}{\partial \sigma_{i}}=\frac{\partial}{\partial \sigma_{i}}\langle\theta| H|\theta\rangle
$$

Understanding the celgorithm
Input: Fixed input state lin, unitury $U$, observable $H$. the eth compenert of the gradient is

$$
\frac{\partial E}{\partial \theta_{i}}=\frac{\partial}{\partial \theta_{i}}\langle\theta| H|\theta\rangle=\frac{\partial}{\partial \sigma_{i}}\langle\operatorname{in}| U_{1}^{\dagger} \ldots U_{p}^{\dagger} H U_{p} \cdots U_{l}|i n\rangle
$$

Understanding the celgorithm
Input: Fixed input state line, unitary $U$, observable $H$. the eth component of the gradient is

$$
\frac{\partial E}{\partial \theta_{i}}=\frac{\partial}{\partial \theta_{i}}\langle\theta| H|\theta\rangle=\frac{\partial}{\partial \sigma_{i}}\langle\operatorname{in}| U_{1}^{+} \ldots U_{p}^{+} H U_{p} \cdots U_{l}|i n\rangle
$$

$$
\text { (provect node) }=\langle\text { in }| v_{1}^{+} \ldots \frac{d v_{i}^{+}}{d \theta_{i}} \ldots u_{p}^{+} H u_{p} \cdots v_{1}|i n\rangle t \text { hic. }
$$

Understanding the celgorithm
Input: Fixed inuet state lins, unitory $U$, observible $H$. the ith compenert of the gradient is

$$
\begin{aligned}
& \frac{\partial E}{\partial \sigma_{i}}=\frac{\partial}{\partial \sigma_{i}}\langle\theta| H|\theta\rangle=\frac{\partial}{\partial \sigma_{i}}\langle\operatorname{in}| U_{1}^{+} \ldots U_{p}^{+} H U_{p} \cdots U_{1}|i n\rangle \\
& \text { (provect ader) }=\langle\text { in }| u_{1}^{+} \ldots \frac{d u_{i}^{+}}{d G_{i}} \cdots u_{p}^{+} H u_{p} \ldots v_{1}|i n\rangle+\text { h.c. } \\
& \left.\left.=2 \operatorname{Re}\langle\operatorname{in}| U_{1}^{+} \ldots U_{p}^{+} H U_{p} \cdots \frac{d U_{i}}{d s_{i}} \cdots U_{1} \right\rvert\, \text { in }\right\rangle
\end{aligned}
$$

Understanding the celgorithm
Input: Fixed inut state lins, unitory $U$, observible $H$. the ith compenert of the gradient is

$$
\begin{aligned}
& \frac{\partial E}{\partial \sigma_{i}}=\frac{\partial}{\partial \sigma_{i}}\langle\theta| H|\theta\rangle=\frac{\partial}{\partial \sigma_{i}}\langle\operatorname{in}| U_{1}^{+} \ldots U_{p}^{\dagger} H U_{p} \cdots U_{1}|i n\rangle \\
& \text { (provect ade) })=\langle\text { in }| u_{1}^{+} \ldots \frac{d u_{i}^{+}}{d G_{i}} \cdots u_{p}^{+} H u_{p} \ldots v_{1}|i n\rangle+\text { h.c. } \\
& \left.\left.=2 R_{e}\langle\operatorname{in}| v_{1}^{+} \ldots U_{p}^{+} H U_{p} \cdots \frac{d U_{i}}{d \theta_{i}} \cdots U_{1} \right\rvert\, \text { in }\right\rangle \\
& \text { Notatin: } \quad \stackrel{\downarrow}{=} 2 \operatorname{Re} \text { prod }\left[U_{p} \cdots U_{1}|i n\rangle, H U_{p} \cdots \frac{d U_{i}}{d \theta_{i}} \cdots v_{1}|i n\rangle\right] \text {. }
\end{aligned}
$$

Understanding the celgorithm

$$
\left.\frac{\partial E}{\partial \theta_{i}}=2 \operatorname{Re} \operatorname{prod}\left[U_{p} \cdots U_{1}|i n\rangle, H U_{p} \cdots \frac{d u_{i}}{d \theta_{i}} \cdots U_{1}| | i n\right\rangle\right]
$$

Understanding the celgorithm

$$
\frac{\partial E}{\partial \theta_{i}}=2 \operatorname{Reprod}\left[H\left(u_{p} \cdots u_{1}\left|i_{n}\right\rangle, u_{p} \cdots \frac{d u_{i}}{d \theta_{i}} \cdots v_{1}|i n\rangle\right]\right.
$$

Understanding the celgorithm

$$
\frac{\partial E}{\partial \theta_{i}}=2 \operatorname{Re} \operatorname{prod}\left[H\left(u_{p} \cdots u_{1}\left|l_{i n}\right\rangle, \quad u_{p} \cdots \frac{d u_{i}}{d \theta_{i}} \cdots v_{1}|i n\rangle\right]\right.
$$

Algorithm: Start with $i=P$

Understanding the celgorithm

$$
\frac{\partial E}{\partial \theta_{i}}=2 \operatorname{Reprod}\left[H u_{p} \cdots u_{1}\left|l_{i n}\right\rangle, u_{p} \cdots \frac{d u_{i}}{d \theta_{i}} \cdots v_{1}|i n\rangle\right]
$$

Algorithm: Start with $i=P$
$\rightarrow$ Compute $2 \operatorname{Re} \operatorname{prod}\left[H\left|p \cdots v_{1}\right|\right.$ in $\rangle, \left.\frac{d U_{p}}{d \epsilon_{p}} U_{p-1} \ldots U_{1} \right\rvert\,$ in $\left.\rangle\right]$

Understanding the celgorithm

$$
\frac{\partial E}{\partial \theta_{i}}=2 \operatorname{Reprod}\left[H\left(U_{p} \cdots v_{1}\left|i_{i n}\right\rangle, U_{p} \cdots \frac{\partial u_{i}}{\partial \theta_{i}} \cdots v_{1}|i n\rangle\right]\right.
$$

Algorithm: Start with $i=P$ O(P) work
$\rightarrow$ Compute 2 Re prod $\left[H\right.$ up $\cdots v_{1} \mid$ in $\rangle, \frac{d U_{p}}{d \epsilon_{p}} U_{p-1} \cdots v_{1}$ lin $\left.\rangle\right]$

Understanding the celgorithm

$$
\frac{\partial E}{\partial \theta_{i}}=2 \operatorname{Reprod}\left[H\left(u_{p} \cdots u_{1}\left|i_{i n}\right\rangle, \quad u_{p} \cdots \frac{d u_{i}}{d \theta_{i}} \cdots v_{1}|i n\rangle\right]\right.
$$

Algorithm: Start with $i=P$
$\rightarrow$ Compute $2 \operatorname{Re} \operatorname{prod}\left[H\left\langle p \cdots v_{1}\right|\right.$ in $\rangle, \frac{d U_{p}}{d \epsilon_{p}} U_{p-1} \cdots v_{1}$ lin $\left.\rangle\right]$ Mar to $i=D-1$

$$
\rightarrow \text { Compute } 2 \text { Re } \operatorname{prod}\left[\begin{array}{l} 
\\
\left.\left.H u_{p} \ldots v_{1}|i n\rangle, U_{p} \frac{d U_{p-1}}{d t_{p-1}} U_{p-2} \cdots v_{1}, \text { in }\right\rangle\right]
\end{array}\right.
$$

Understanding the celgorithm

$$
\frac{\partial E}{\partial \theta_{i}}=2 \operatorname{Reprod}\left[H\left(u_{p} \cdots u_{1}\left|i_{i n}\right\rangle, \quad u_{p} \cdots \frac{d u_{i}}{d \theta_{i}} \cdots v_{1}|i n\rangle\right]\right.
$$

Algorithm: Start with $i=P$
$\rightarrow$ Compute $2 \operatorname{Re} \operatorname{prod}\left[H U p \cdots v_{1} \mid\right.$ in $\rangle, \left.\frac{d U_{p}}{d \epsilon_{p}} U_{p-1} \cdots U_{1} \right\rvert\,$ in $\left.\rangle\right]$ Mare to $i=D-1$

$$
\left.\rightarrow \text { Compute } 2 \text { Re } \operatorname{prod}\left[H u_{p} \ldots v_{1}|i n\rangle, U_{p} \frac{d u_{p-1}}{d t_{p-1}} U_{p-2} \cdots v_{1}, l i n\right\rangle\right]
$$

$$
\left.\left.=2 \operatorname{le} \operatorname{prod}\left[U_{p}^{+} H U_{p} \cdots v_{1} l \text { in }\right\rangle, \frac{d U_{p-1}}{d s_{p-1}} U_{p-1}^{+} u_{p-1} u_{p-2} \cdots v_{1} l i n\right\rangle\right]
$$

Understanding the celgorithm

$$
\frac{\partial E}{\partial \theta_{i}}=2 \operatorname{Reprod}\left[H\left(u_{p} \cdots v_{1}\left|i_{i n}\right\rangle, \quad u_{p} \cdots \frac{d u_{i}}{d \theta_{i}} \cdots v_{1}|i n\rangle\right]\right.
$$

Algorithm: Start with $i=P$
$\rightarrow$ Compute $2 \operatorname{Re} \operatorname{prod}\left[H U p \cdots v_{1} \mid\right.$ in $\geqslant, \left.\frac{d U_{p}}{d \epsilon_{p}} U_{p-1} \cdots U_{1} \right\rvert\,$ in $\left.\rangle\right]$ Mare to $i=D-1$

$$
\begin{aligned}
& \text { Move to } i=P-1 \\
& \left.\left.\rightarrow \text { Compute } 2 \text { Re } \operatorname{prod}\left[H u_{p} \ldots v_{1} l i n\right\rangle, U_{p} \frac{d u_{p-1}}{d k_{p-1}} U_{p-2} \cdots v_{1} l i n\right\rangle\right]
\end{aligned}
$$

$$
\left.\left.=2 \operatorname{le} \operatorname{prod}\left[U_{p}^{+} H U_{p} \cdots U_{1} \mid \text { in }\right\rangle, \left.\frac{d U_{p-1}}{d E_{p-1}} U_{p-1}^{+} U_{p-1} U_{p-2} \cdots v_{1} \right\rvert\, \text { in }\right\rangle\right]
$$

Understanding the celgorithm

$$
\frac{\partial E}{\partial \theta_{i}}=2 \operatorname{Reprod}\left[H\left(u_{p} \cdots v_{1}\left|i_{i n}\right\rangle, \quad u_{p} \cdots \frac{d u_{i}}{d \theta_{i}} \cdots v_{1}|i n\rangle\right]\right.
$$

Algorithm: Start with $i=P$

$$
O(P) \text { work }
$$

$\rightarrow$ Compute $2 \operatorname{Re} \operatorname{prod}\left[H U p \cdots v_{1}|i n\rangle, \frac{d U_{p}}{d E_{p}} U_{p-1} \cdots U_{1}\right.$, in $\left.\rangle\right]$ Mare to $i=D-1$

$$
\left.=2 \operatorname{le} \operatorname{prod}\left[U_{p}^{+} H U_{p} \cdots U_{1} \mid \text { in }\right\rangle, \frac{d U_{p-1}}{d s_{p-1}} U_{p-1}^{+} U_{p-1} U_{p-2} \cdots v_{1}|\operatorname{lin}\rangle\right]
$$

$O(1)$ work

Understanding the celgorithm

$$
\left.\left.\frac{\partial E}{\partial \epsilon_{p-1}}=2 \operatorname{ke} \operatorname{prad}\left[U_{p}^{+} H U_{p} \cdots U_{1} \mid \text { in }\right\rangle, \left.\frac{d U_{p-1}}{d s_{p-1}} U_{p-1}^{+} U_{p-1} U_{p-2} \cdots v_{1} \right\rvert\, \text { in }\right\rangle\right]
$$

Iterate this:

Understanding the celgorithm

$$
\left.\left.\frac{\partial E}{\partial \epsilon_{p-1}}=2 \operatorname{le} \operatorname{prod}\left[U_{p}^{+} H U_{p} \cdots v_{1} \mid \text { in }\right\rangle, \frac{d U_{p-1}}{d s_{p-1}} U_{p-1}^{+} U_{p-1} u_{p-2} \cdots v_{1} \text { lin }\right\rangle\right]
$$

Iterate this:

$$
\begin{aligned}
& \text { Iterate thin: } \\
& \left.\frac{\partial E}{\partial \theta_{p-2}}=2 \text { le prod }\left[U_{p-1}+U_{p}^{+} H\left(u_{p} \ldots v_{1} \text { lin }\right\rangle, \frac{\partial U_{p-2}}{d \theta_{p-2}} u_{p-2}^{+} U_{p}^{+}+u_{p-1} v_{p-2} \ldots v_{1} \text { in }\right\rangle\right]
\end{aligned}
$$

Understanding the celgorithm

$$
\left.\left.\frac{\partial E}{\partial \epsilon_{p-1}}=2 \operatorname{le} \operatorname{prod}\left[U_{p}^{+} H U_{p} \cdots v_{1} \mid \text { in }\right\rangle, \frac{d U_{p-1}}{d s_{p-1}} U_{p-1}^{+} U_{p-1} U_{p-2} \cdots v_{1} \text { lin }\right\rangle\right]
$$

Iterate this:

$$
\begin{aligned}
& \text { Iterate this: } \\
& \left.\frac{\partial E}{\partial G_{p-2}}=2 \text { he prod }\left[U_{p-1}^{+}+U_{p}^{+} H\left(u_{p} \cdots v_{1} l i n\right\rangle, \frac{\partial U_{p-2}}{d \theta_{p-2}} u_{p-2}^{+} U_{p-1}^{+} U_{p, 1} u_{p-2} \cdots u_{1} \text { lin }\right\rangle\right]
\end{aligned}
$$

Again we have $O(1)$ work in this Step

Understanding the celgorithm

$$
\left.\left.\frac{\partial E}{\partial \epsilon_{p-1}}=2 \operatorname{le} \operatorname{prod}\left[U_{p}^{+} H U_{p} \cdots v_{1} \mid \text { in }\right\rangle, \left.\frac{d U_{p-1}}{d s_{p-1}} U_{p-1}^{+} U_{p-1} u_{p-2} \cdots v_{1} \right\rvert\, \text { in }\right\rangle\right]
$$

Iterate this:

$$
\left.\frac{\partial E}{\partial G_{p-2}}=2 \operatorname{le} \operatorname{prod}\left[U_{p-1}^{+} U_{p}^{+} H\left(u_{p} \ldots u_{1} \text { lin }\right\rangle, \frac{\partial U_{p-2}}{d \theta_{p-2}} u_{p-2}^{+} U_{p}^{+}+u_{p-1}, v_{p-2} \ldots v_{1} \operatorname{lin}\right\rangle\right]
$$

Again we have $O(1)$ work in this seep
Thus, this "reverse mark" algorithm has:

Understanding the celgorithm

$$
\left.\left.\frac{\partial E}{\partial s_{p-1}}=2 \operatorname{le} \operatorname{prod}\left[U_{p}^{+} H U_{p} \cdots v_{1} \mid \text { in }\right\rangle, \frac{d U_{p-1}}{d s_{p-1}} U_{p-1}^{+} u_{p-1} u_{p-2} \cdots v_{1} \text { lin }\right\rangle\right]
$$

Iterate this:

$$
\left.\frac{\partial E}{\partial G_{p-2}}=2 \text { he prod }\left[U_{p-1}+U_{p}^{+} H u_{p} \cdots v_{1}|i n\rangle, \frac{\partial U_{p-2}}{\partial \partial_{p-2}} u_{p-2}^{+} U_{p-1}^{+} u_{p-1}, u_{p-2} \cdots v_{1} \text { lin }\right\rangle\right]
$$

Again we have $O(1)$ work in this Step
Thus, this "reverse mode" algorithm has:

- $O(P)$ work in the list step [Compute $\frac{\partial E}{\partial G_{p}}$ ]

Understanding the celgorithm

$$
\left.\left.\frac{\partial E}{\partial s_{p-1}}=2 \operatorname{le} \operatorname{prod}\left[U_{p}^{+} H U_{p} \cdots v_{1} \mid \text { in }\right\rangle, \frac{d U_{p-1}}{d s_{p-1}} U_{p-1}^{+} u_{p-1} u_{p-2} \cdots v_{1} \text { lin }\right\rangle\right]
$$

Iterate this:

$$
\left.\frac{\partial E}{\partial G_{p-2}}=2 \text { he prod }\left[U_{p-1}+U_{p}^{+} H u_{p} \cdots v_{1}|i n\rangle, \frac{\partial U_{p-2}}{\partial \partial_{p-2}} u_{p-2}^{+} U_{p-1}^{+} u_{p-1}, u_{p-2} \cdots v_{1} \text { lin }\right\rangle\right]
$$

Again we have $O(1)$ work in this step
Thus, this "reverse mode" algorithm has:

- $O(P)$ work in the list step [Compute $\frac{\partial E}{\partial G_{p}}$ ]
- $O(1)$ work for the remaining $P-1$ steps

Understanding the celgorithm

$$
\left.\left.\frac{\partial E}{\partial s_{p-1}}=2 \operatorname{le} \operatorname{prod}\left[U_{p}^{+} H U_{p} \cdots v_{1} \mid \text { in }\right\rangle, \frac{d U_{p-1}}{d s_{p-1}} U_{p-1}^{+} u_{p-1} u_{p-2} \cdots v_{1} \text { lin }\right\rangle\right]
$$

Iterate thin:

$$
\left.\left.\frac{\partial E}{\partial G_{p-2}}=2 \operatorname{le} \text { prod }\left[U_{p-1}+U_{p}^{+} H u_{p} \cdots u_{1} l i n\right\rangle, \frac{\partial U_{p-2}}{d \theta_{p-2}} u_{p-2}^{+} U_{p-1}^{+} u_{p-1}, u_{p-2} \cdots u_{1} \text { lin }\right\rangle\right]
$$

Again we have $O(1)$ work in this step
Thus, this "reverse mode" algorithm has:

- $O(P)$ work in the list step [Compute $\frac{\partial E}{\partial G_{p}}$ ]
- O(1) work for the remaining P-1 steps $\Rightarrow O(D)$ total work.

Algorithm 1: Calculating the noise-free gradient with state-vectors, using "reverse mode". Let $G$ be the complexity of effecting a fixed-size gate upon an $N$-qubit state-vector. Typically $G$ scales with the number of amplitudes in the state-vector as $G=\mathcal{O}\left(2^{N}\right)$.

Input : State-vectors $|\lambda\rangle,|\phi\rangle,|\mu\rangle$, an immutable input state $\mid$ in $\rangle$, some representation of a circuit $U_{1: P}$ with a single unique parameter in each gate, and a Hamiltonian $\hat{H}$ in any applicable representation
Output: Each element of $\nabla\langle E\rangle$

```
\(|\lambda\rangle:=\mid\) in \(\rangle\)
\(|\lambda\rangle \leftarrow \hat{U}_{1: P}|\lambda\rangle\)
\(|\phi\rangle:=|\lambda\rangle\)
\(|\lambda\rangle \leftarrow \hat{H}|\lambda\rangle\)
// clone state in \(\mathcal{O}(G)\)
// apply \(P\) gates in \(\mathcal{O}(P G)\)
    // clone state in \(\mathcal{O}(G)\)
    // apply \(\hat{H}\) in \(\mathcal{O}(h N G)\)
for \(i \in\{P, \ldots, 1\}\) do
    \(|\phi\rangle \leftarrow \hat{U}_{i}^{\dagger}|\phi\rangle \quad\) // apply gate in \(\mathcal{O}(G)\)
    \(|\mu\rangle:=|\phi\rangle \quad\) // clone state in \(\mathcal{O}(G)\)
    \(|\mu\rangle \leftarrow\left(\mathrm{d} \hat{U}_{i} / \mathrm{d} \theta_{i}\right)|\mu\rangle \quad / /\) apply non-unitary in \(\mathcal{O}(G)\)
    \(\nabla\langle E\rangle_{i}=2 \Re\langle\lambda \mid \mu\rangle \quad / /\) compute inner product in \(\mathcal{O}(G)\)
    if \(i>1\) then
        \(|\lambda\rangle \leftarrow U_{i}^{\dagger}|\lambda\rangle \quad\) // apply gate in \(\mathcal{O}(G)\)
    end
end
```


## The algorithm

$\left.2 \operatorname{le} \operatorname{prod}\left[H\left|p \cdots v_{1}\right| i n\right\rangle, \frac{d v_{p}}{d \epsilon_{p}} u_{p-1} \ldots v_{1}|i n\rangle\right]$


Algorithm 1: Calculating the noise-free gradient with state-vectors, using "reverse mode". Let $G$ be the complexity of effecting a fixed-size gate upon an $N$-quit state-vector. Typically $G$ scales with the number of amplitudes in the state-vector as $G=\mathcal{O}\left(2^{N}\right)$.
Input : State-vectors $|\lambda\rangle,|\phi\rangle,|\mu\rangle$, an immutable input state $\mid$ in $\rangle$, some representation of a circuit $U_{1: P}$ with a single unique parameter in each gate, and a Hamiltonian $\hat{H}$ in any applicable representation
Output: Each element of $\nabla\langle E\rangle$

```
\(|\lambda\rangle:=\mid\) in \(\rangle\)
\(|\lambda\rangle \leftarrow \hat{U}_{1: P}|\lambda\rangle\)
\(|\phi\rangle:=|\lambda\rangle\)
\(|\lambda\rangle \leftarrow \hat{H}|\lambda\rangle\)
// clone state in \(\mathcal{O}(G)\)
// apply \(P\) gates in \(\mathcal{O}(P G)\)
    // clone state in \(\mathcal{O}(G)\)
for \(i \in\{P, \ldots, 1\}\) do
    \(|\phi\rangle \leftarrow \hat{U}_{i}^{\dagger}|\phi\rangle \quad\) // apply gate in \(\mathcal{O}(G)\)
    \(|\mu\rangle:=|\phi\rangle \quad\) // clone state in \(\mathcal{O}(G)\)
    \(|\mu\rangle \leftarrow\left(\mathrm{d} \hat{U}_{i} / \mathrm{d} \theta_{i}\right)|\mu\rangle \quad / /\) apply non-unitary in \(\mathcal{O}(G)\)
    \(\nabla\langle E\rangle_{i}=2 \Re\langle\lambda \mid \mu\rangle \quad / /\) compute inner product in \(\mathcal{O}(G)\)
    if \(i>1\) then
        \(|\lambda\rangle \leftarrow U_{i}^{\dagger}|\lambda\rangle \quad\) // apply gate in \(\mathcal{O}(G)\)
    end
end
```


## The algorithm

$\left.2 \operatorname{le} \operatorname{prod}\left[H\left|p \cdot \cdots v_{1}\right| i n\right\rangle, \frac{d v_{p}}{d \epsilon_{p}} u_{p-1} \ldots v_{1}|i n\rangle\right]$

Algorithm 1: Calculating the noise-free gradient with state-vectors, using "reverse mode". Let $G$ be the complexity of effecting a fixed-size gate upon an $N$-quit state-vector. Typically $G$ scales with the number of amplitudes in the state-vector as $G=\mathcal{O}\left(2^{N}\right)$.
Input : State-vectors $|\lambda\rangle,|\phi\rangle,|\mu\rangle$, an immutable input state $\mid$ in $\rangle$, some representation of a circuit $U_{1: P}$ with a single unique parameter in each gate, and a Hamiltonian $\hat{H}$ in any applicable representation
Output: Each element of $\nabla\langle E\rangle$

```
\(|\lambda\rangle:=\mid\) in \(\rangle\)
\(|\lambda\rangle \leftarrow \hat{U}_{1: P}|\lambda\rangle\)
\(|\phi\rangle:=|\lambda\rangle\)
\(|\lambda\rangle \leftarrow \hat{H}|\lambda\rangle\)
// clone state in \(\mathcal{O}(G)\)
// apply \(P\) gates in \(\mathcal{O}(P G)\)
    // clone state in \(\mathcal{O}(G)\)
    // apply \(\hat{H}\) in \(\mathcal{O}(h N G)\)
for \(i \in\{P, \ldots, 1\}\) do
    \(|\phi\rangle \leftarrow \hat{U}_{i}^{\dagger}|\phi\rangle \quad\) // apply gate in \(\mathcal{O}(G)\)
    \(|\mu\rangle:=|\phi\rangle \quad\) // clone state in \(\mathcal{O}(G)\)
    \(|\mu\rangle \leftarrow\left(\mathrm{d} \hat{U}_{i} / \mathrm{d} \theta_{i}\right)|\mu\rangle\)
    // apply non-unitary in \(\mathcal{O}(G)\)
    \(\nabla\langle E\rangle_{i}=2 \Re\langle\lambda \mid \mu\rangle\)
// compute inner product in \(\mathcal{O}(G)\)
    if \(i>1\) then
        \(|\lambda\rangle \leftarrow U_{i}^{\dagger}|\lambda\rangle \quad\) // apply gate in \(\mathcal{O}(G)\)
    end
end
```

Some benchmarks


Circuit A


Circuit B


Circuit C

for different circuit consatze (why?)

Tine to compute $D E$ for "reverse mate" - O(P) alpo and "reference" - $O\left(p^{2}\right)$ alp.

## Some extersions

## (1) Gates wl multiple parameters

```
Algorithm 3: A replacement of lines 6-9 in Algorithm 1 to handle a gate \(\hat{U}_{j}\) with
multiple parameters, \(\phi_{1}, \ldots, \phi_{n}\), which correspond to gradient elements with in-
dices \(k_{1}, \ldots, k_{n}\)
// loop over each parameter in gate \(\hat{U}_{i}\)
\(\mathbf{1}\) for \(j \in\{1, \ldots, n\}\) do
\(2||\mu\rangle:=| \phi\rangle \quad\) // clone state in \(\mathcal{O}(G)\)
\(3 \quad|\mu\rangle \leftarrow\left(\mathrm{d} \hat{U}_{i} / \mathrm{d} \phi_{j}\right)|\mu\rangle \quad / /\) apply non-unitary in \(\mathcal{O}(G)\)
\(4 \quad \nabla\langle E\rangle_{k_{j}}=2 \Re\langle\lambda \mid \mu\rangle \quad\) // compute inner product in \(\mathcal{O}(G)\)
5 end
```

Some extersions
(2) Repeated parameters (e.g., Q1OA)

- Make each parareter unizue
- Run reverse-made celgorithm
- Combire results

Some extersions
(2) Repeated parameters (e.g., Q1OA)

- Make each parareter unizue
- Run reverse-made algorithin
- Combine results

Eg, say $O$ appecrs in gace $U_{i}+U_{S}$. Thar,

$$
\begin{aligned}
\frac{\partial E}{\partial \theta}= & 2 \operatorname{le} \operatorname{prod}\left[U|i n\rangle, H u_{p} \cdots \frac{d u_{1}}{d \epsilon} \cdots u_{1}|i n\rangle\right]+ \\
& 2 \operatorname{le} \operatorname{prod}\left[u|i n\rangle, H u p \cdots \frac{d u_{i}}{d \theta} \ldots u_{1}|i n\rangle\right] .
\end{aligned}
$$

Some extensions
(3) Non-unitory gates

- Le never used cony property of unitaries w/ $U_{1} \cdots U_{p}$.

Some extensions
(3) Non-unitory gates

- We never used cony property of unitaries w/ $U_{1} \cdots U_{p}$.
$\Rightarrow$ The same algorithm works $\omega /$ invertible matrices $M_{1}, \ldots, M_{p}$.

Some extersions
(3) Non-unitery gates

- We never used cony property of unitaries w/ $U_{1} \cdots U_{p}$.
$\Rightarrow$ The same algorithm works $\omega /$ invertible matrices $M_{1}, \ldots, M_{p}$.
* Just recall that $M_{i}^{-1} \neq M_{i}^{+}$and be careful to use each appropriately

Some extensions
(3) Non-unitory gates

- We never used cony property of unitaries w/ $U_{1} \ldots U_{p}$.
$\Rightarrow$ The same algorithnen works $\omega /$ invertible matrices $M_{1}, \ldots, M_{p}$. * Just recall that $M_{i}^{-1} \neq M_{i}^{+}$and be careful to use each appropriately

$$
\left.\frac{\partial E}{\partial G_{p-1}}=2 \operatorname{le} \operatorname{prod}\left[M_{p-1}^{+} M_{p}^{+} H_{\rho} \ldots \mu_{1}|i\rangle\right\rangle, \frac{\partial M_{p-2}}{\partial \theta_{p-2}} M_{p-2}^{-1} M_{p-1}^{-1} \mu_{p-1} \mu_{p-2} \ldots \mu_{1}|i n\rangle\right]
$$

Some extersions
(4) Non-hermitian operators $A$

Reverse mode algo works w/ minor modifications:

$$
\frac{\partial E}{\partial \theta_{i}}=\left\langle\operatorname{in} \left\lvert\, \frac{\partial U^{+}}{\partial \theta_{i}} A U \operatorname{lin}\right.\right\rangle+\langle\operatorname{in}| U^{+} A \frac{\partial U}{\partial \theta_{i}}|\operatorname{in}\rangle
$$

(This leack to compex gradiones)

Some extersions
(5) Noisy Simulation
(a) W/ inuotible channels
(b) Using Soperoperators + vectorization (Choi-Jamidtrassi samo-phisn)

Other extensias they don't mention

- Tire evelution wi tensar retworks e.g., MPS, TTU, ...

Main Result

* (Description of) an O(P) algorithm for the gradient DE...
...Under these cossumptions:
(1) State vector simulator
- ie, we have access to the foll wavefunction.
(2) Ability to apply the observille $H$ to a state $|\psi\rangle$
- ie, we can compute HIS
(3) Ability to compute gate derivatives and apply then to states
- ie, $\frac{d v_{i}}{d \theta_{i}}$ and $\frac{d v_{i}}{d \theta_{i}}|\psi\rangle$

