Quantum walks

Daniel J. Bernstein
University of Illinois at Chicago

Focusing on quantum walks
as an algorithm-design tool:
e.g. Grover's algorithm.
e.g. Ambainis's algorithm.

Can also study quantum walks
on much more general graphs.
2008 Childs, 2009 Lovett–
Cooper–Everitt–Trevers–Kendon:
Can view, e.g., Shor's algorithm
as quantum walk on Shor graph.

Examples of applications to crypto

Minimum asymptotic ops known, assuming plausible heuristics:

pre-q	post-q	problem
1	0.5	cipher
ρ	$\rho/2$	McEliece
$0.791\ldots$	$0.462\ldots$	MQ
$0.290\ldots$	$0.241\ldots$	subset sum

"Pre-q" e: as $n \to \infty$, $2^{(e+o(1))n}$ simple non-quantum ops.

"Post-q" e: as $n \to \infty$, $2^{(e+o(1))n}$ simple quantum ops.

"Cipher": find n-bit cipher key.
0.5: 1996 Grover.

"McEliece": in linear code of length $(1 + o(1))n \log_2 n$ and dimension $(R + o(1))n \log_2 n$, decode $(1 - R + o(1))n$ errors.

“McEliece”: in linear code of length $(1 + o(1))n \log_2 n$ and dimension $(R + o(1))n \log_2 n$, decode $(1 - R + o(1))n$ errors.

$\rho = (1 - R) \log_2(1/(1 - R))$: 1962 Prange.

“McEliece”: in linear code of length $(1 + o(1))n \log_2 n$ and dimension $(R + o(1))n \log_2 n$, decode $(1 - R + o(1))n$ errors.

$\rho = (1 - R)\log_2(1/(1 - R))$: 1962 Prange.

$\rho/2$: 2009 Bernstein (via Grover).

“McEliece”: in linear code of length $(1 + o(1))n \log_2 n$ and dimension $(R + o(1))n \log_2 n$, decode $(1 - R + o(1))n$ errors.
$\rho = (1 - R) \log_2(1/(1 - R))$: 1962 Prange.
$\rho/2$: 2009 Bernstein (via Grover).

“MQ”: solve system of n deg-2 equations in n variables over $\mathbf{F}_2$.

"McEliece": in linear code of length $(1 + o(1))n \log_2 n$ and dimension $(R + o(1))n \log_2 n$, decode $(1 - R + o(1))n$ errors.
$\rho = (1 - R)\log_2(1/(1 - R))$:
1962 Prange.
$\rho/2$: 2009 Bernstein (via Grover).

"MQ": solve system of n deg-2 equations in n variables over $\mathbf{F}_2$.
0.791 (modulo calculation errors):
2004 Yang–Chen–Courtois.

"McEliece": in linear code of length $(1 + o(1))n \log_2 n$ and dimension $(R + o(1))n \log_2 n$, decode $(1 - R + o(1))n$ errors.
$\rho = (1 - R) \log_2(1/(1 - R))$: 1962 Prange.
$\rho/2$: 2009 Bernstein (via Grover).

"MQ": solve system of n deg-2 equations in n variables over $\mathbf{F}_2$.
0.791 (modulo calculation errors): 2004 Yang–Chen–Courtois.
0.462: 2017 Bernstein–Yang (via Grover), independently 2017 Faugère–Horan–Kahrobaei–Kaplan–Kashefi–Perret.

"Subset sum" ("hard" case):
find $S \subseteq \{1, 2, \ldots, n\}$ given
$x_1, x_2, \ldots, x_n \in \{0, 1, \ldots, 2^n - 1\}$
and $\sum_{i \in S} x_i$.

"Subset sum" ("hard" case):
find $S \subseteq \{1, 2, \ldots, n\}$ given
$x_1, x_2, \ldots, x_n \in \{0, 1, \ldots, 2^n - 1\}$
and $\sum_{i \in S} x_i$.

0.5: easy.

"Subset sum" ("hard" case): find $S \subseteq \{1, 2, \ldots, n\}$ given $x_1, x_2, \ldots, x_n \in \{0, 1, \ldots, 2^n - 1\}$ and $\sum_{i \in S} x_i$.

0.5: easy.

0.337: 2010 Howgrave-Graham–Joux. Claimed 0.311; error discovered by May–Meurer.

"Subset sum" ("hard" case):
find $S \subseteq \{1, 2, \ldots, n\}$ given
$x_1, x_2, \ldots, x_n \in \{0, 1, \ldots, 2^n - 1\}$
and $\sum_{i \in S} x_i$.

0.5: easy.

0.337: 2010 Howgrave-Graham–Joux. Claimed 0.311; error discovered by May–Meurer.

0.291: 2011 Becker–Coron–Joux.

"Subset sum" ("hard" case): find $S \subseteq \{1, 2, \ldots, n\}$ given $x_1, x_2, \ldots, x_n \in \{0, 1, \ldots, 2^n - 1\}$ and $\sum_{i \in S} x_i$.

0.5: easy.

0.337: 2010 Howgrave-Graham–Joux. Claimed 0.311; error discovered by May–Meurer.

0.291: 2011 Becker–Coron–Joux.

0.241: 2013 Bernstein–Jeffery–Lange–Meurer, using HGJ and quantum walks (not just Grover).

Grover's algorithm

Assume: unique $s \in \{0,1\}^n$ has $f(s) = 0$.

Traditional algorithm to find s: compute f for many inputs, hope to find output 0.
Success probability is very low until #inputs approaches 2^n.

Grover's algorithm

Assume: unique $s \in \{0,1\}^n$
has $f(s) = 0$.

Traditional algorithm to find s:
compute f for many inputs,
hope to find output 0.
Success probability is very low
until #inputs approaches 2^n.

Grover's algorithm takes only $2^{n/2}$
reversible computations of f.
Typically: reversibility overhead
is small enough that this easily
wins for all sufficiently large n.

Start from uniform superposition a over $q \in \{0,1\}^n$: $a_q = 2^{-n/2}$.

Start from uniform superposition a over $q \in \{0,1\}^n$: $a_q = 2^{-n/2}$.

Step 1: Set $a \leftarrow b$ where
$b_q = -a_q$ if $f(q) = 0$,
$b_q = a_q$ otherwise.
This is fast.

Start from uniform superposition a over $q \in \{0,1\}^n$: $a_q = 2^{-n/2}$.

Step 1: Set $a \leftarrow b$ where
$b_q = -a_q$ if $f(q) = 0$,
$b_q = a_q$ otherwise.
This is fast.

Step 2: "Grover diffusion".
Negate a around its average.
This is also fast.

Start from uniform superposition
a over $q \in \{0,1\}^n$: $a_q = 2^{-n/2}$.

Step 1: Set $a \leftarrow b$ where
$b_q = -a_q$ if $f(q) = 0$,
$b_q = a_q$ otherwise.
This is fast.

Step 2: "Grover diffusion".
Negate a around its average.
This is also fast.

Repeat Step 1 + Step 2
about $0.58 \cdot 2^{n/2}$ times.

Start from uniform superposition a over $q \in \{0,1\}^n$: $a_q = 2^{-n/2}$.

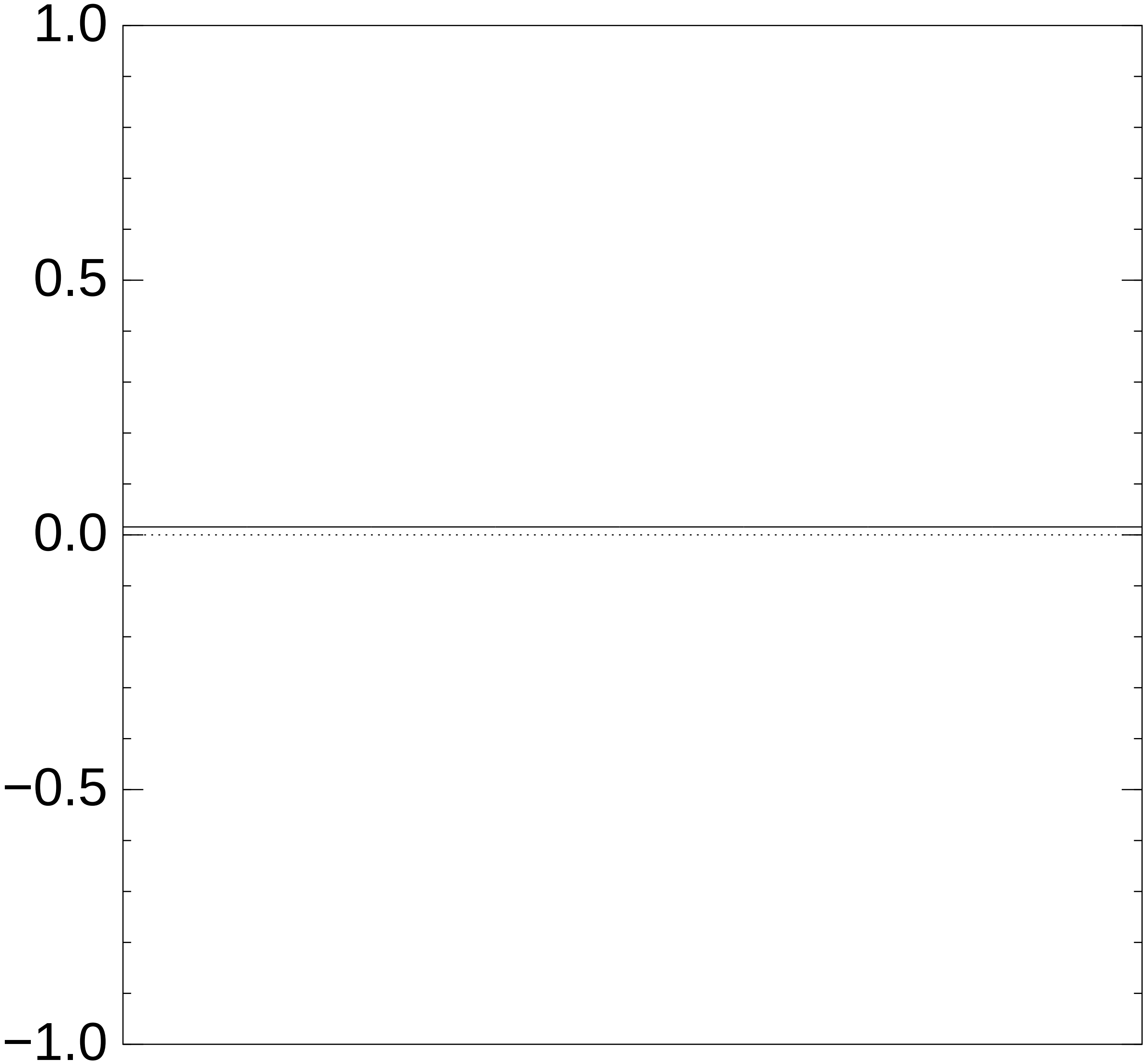
Step 1: Set $a \leftarrow b$ where
$b_q = -a_q$ if $f(q) = 0$,
$b_q = a_q$ otherwise.
This is fast.

Step 2: "Grover diffusion".
Negate a around its average.
This is also fast.

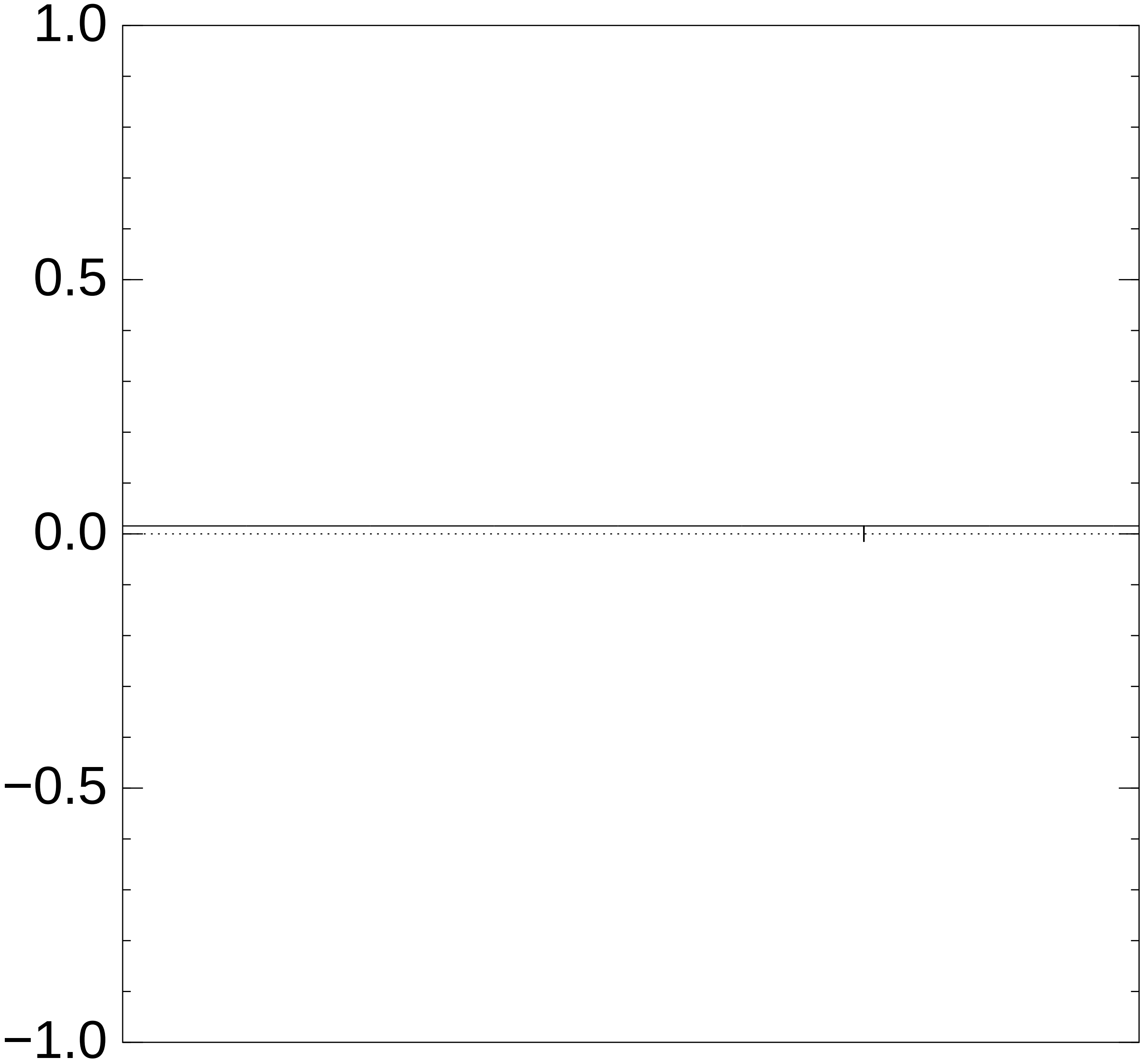
Repeat Step 1 + Step 2
about $0.58 \cdot 2^{n/2}$ times.

Measure the n qubits.
With high probability this finds s.

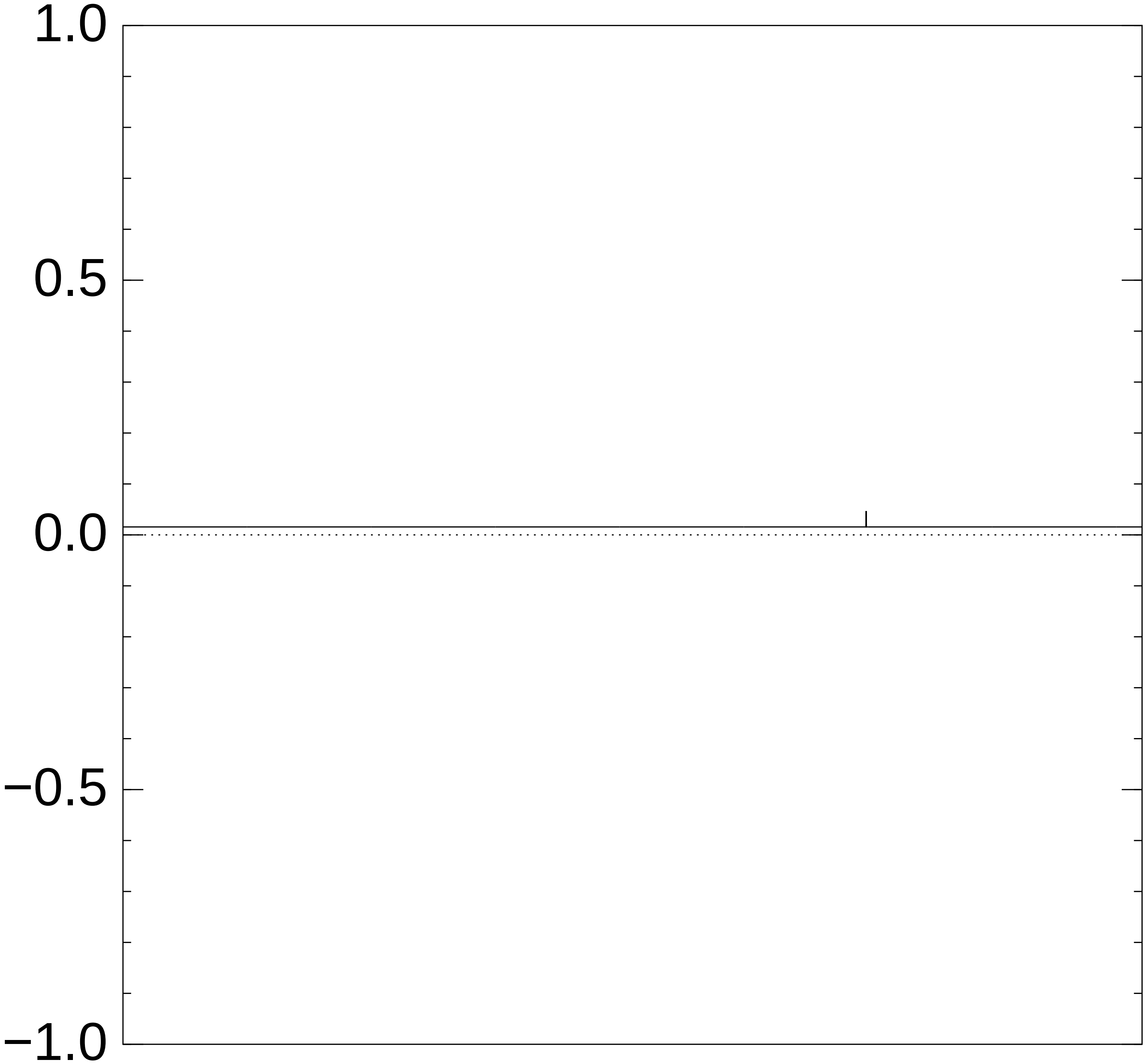
Normalized graph of $q \mapsto a_q$
for an example with $n = 12$
after 0 steps:



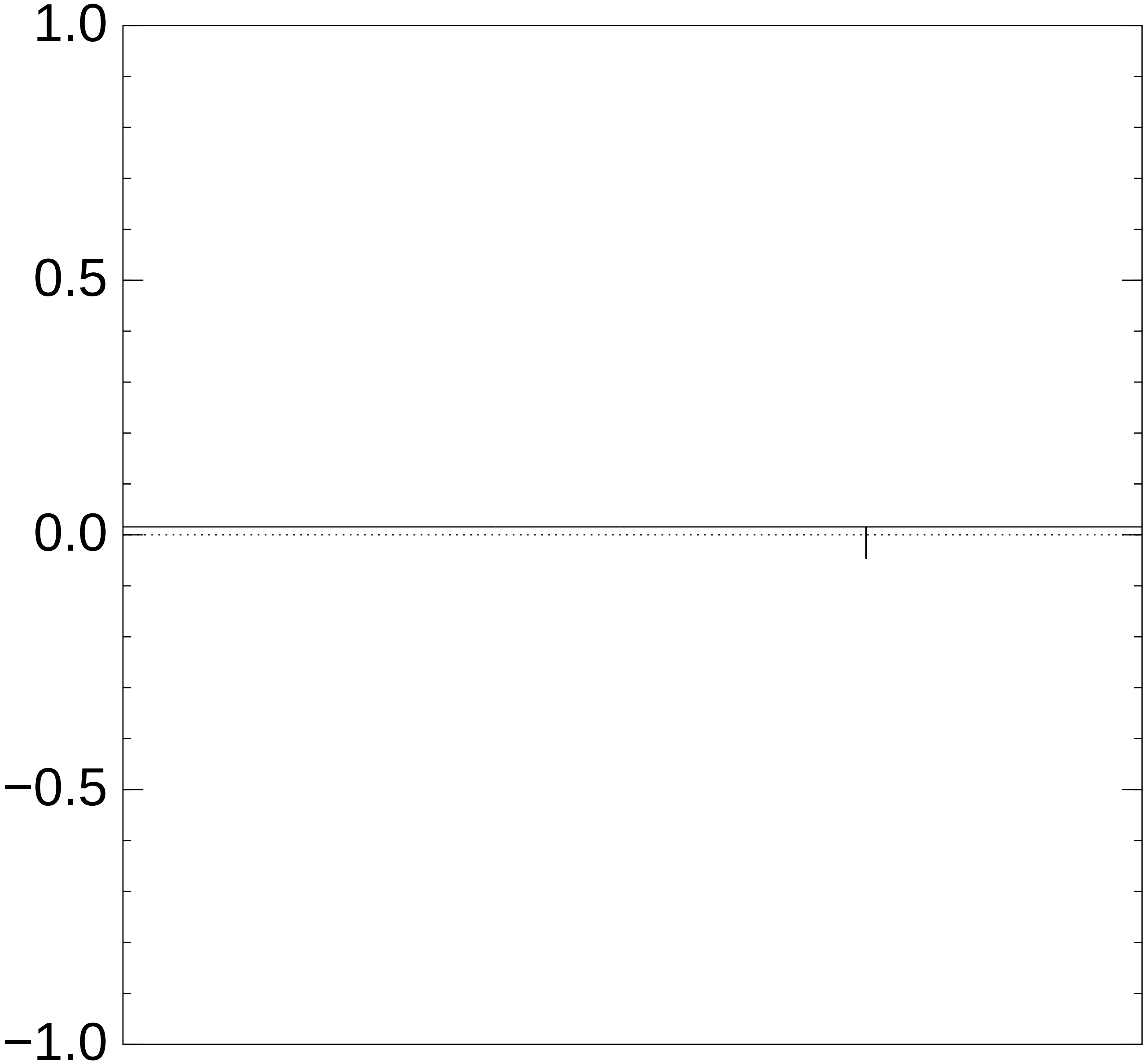
Normalized graph of $q \mapsto a_q$ for an example with $n = 12$ after Step 1:



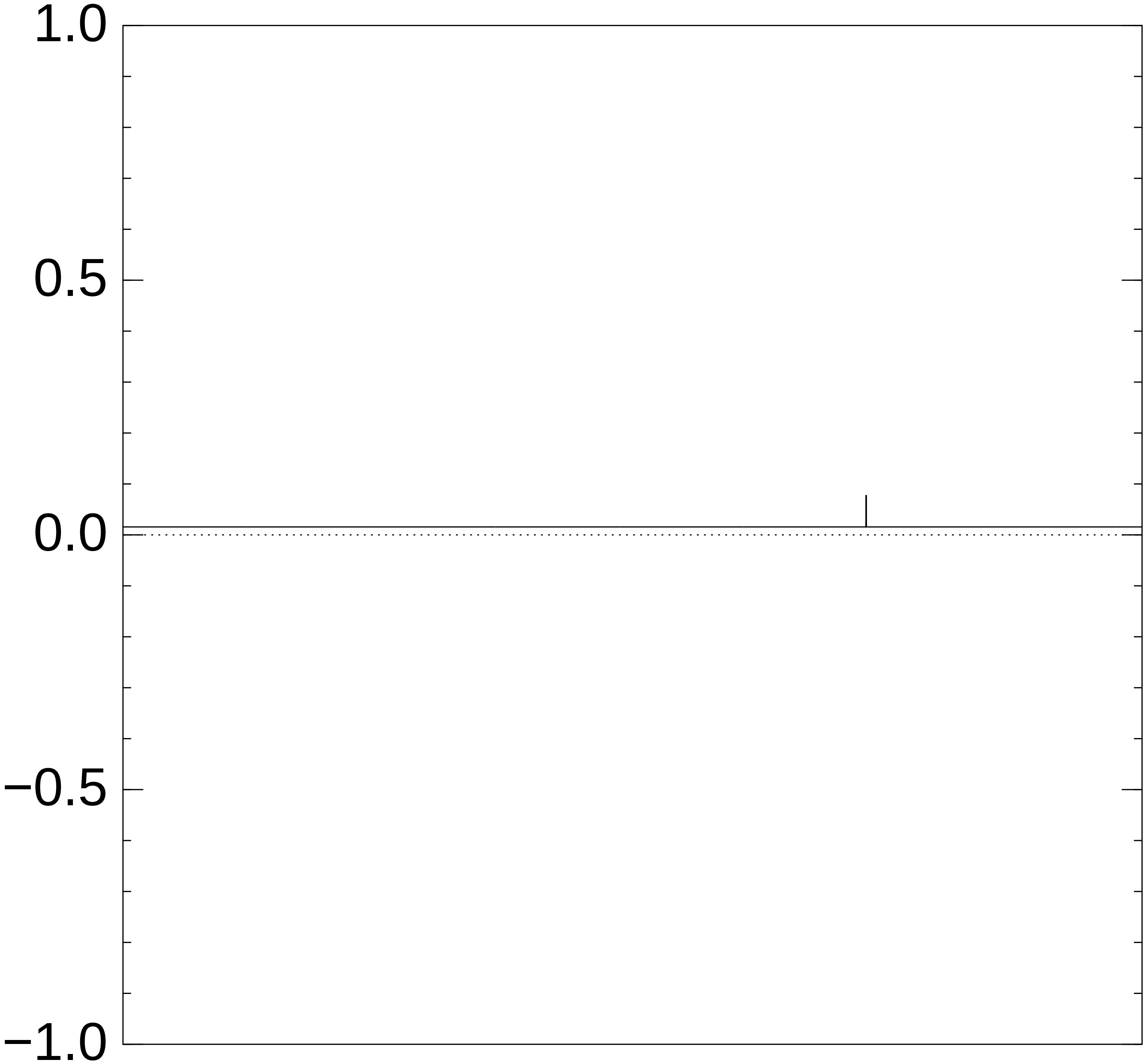
Normalized graph of $q \mapsto a_q$
for an example with $n = 12$
after Step 1 + Step 2:



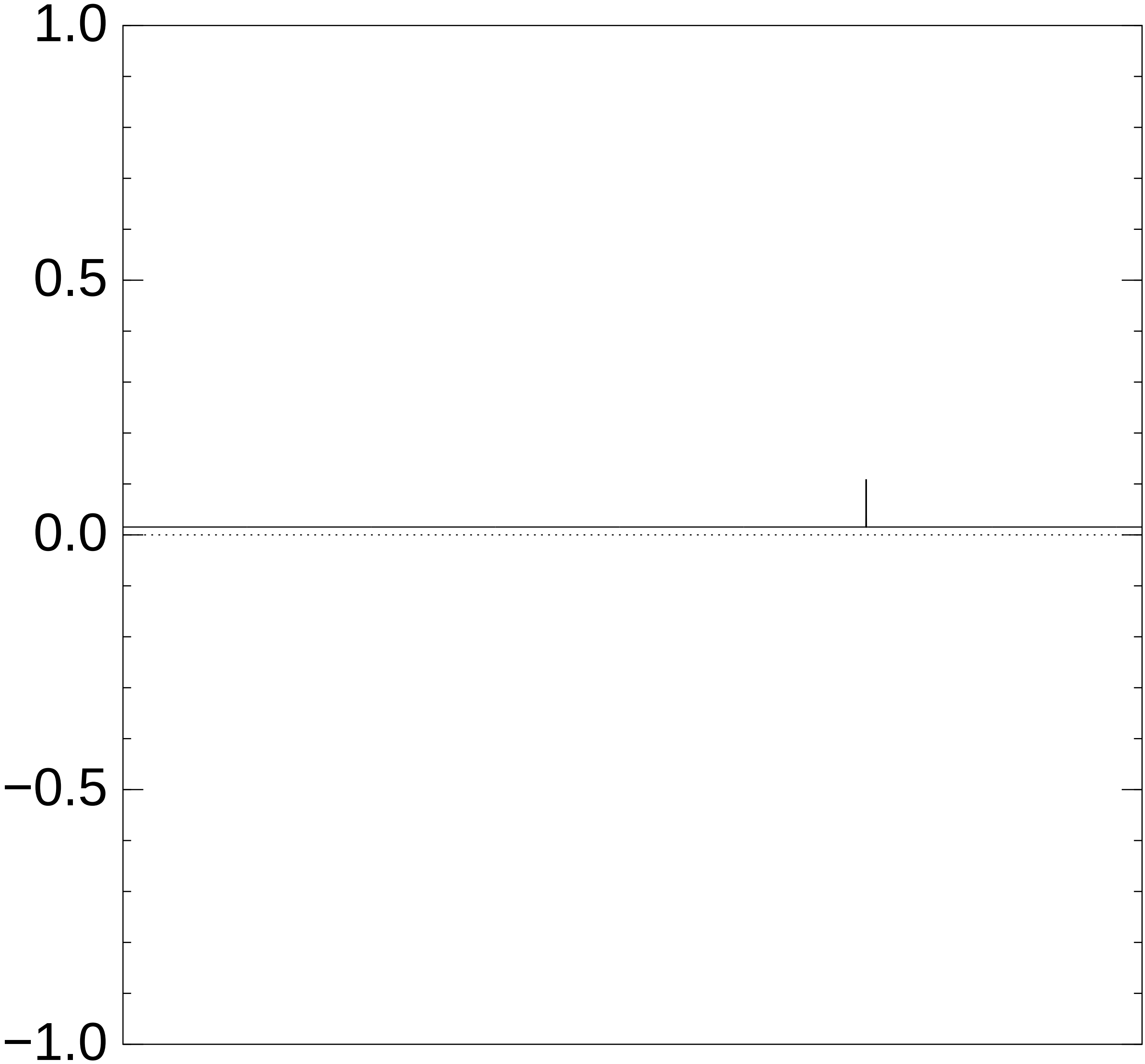
Normalized graph of $q \mapsto a_q$ for an example with $n = 12$ after Step $1 +$ Step $2 +$ Step 1:



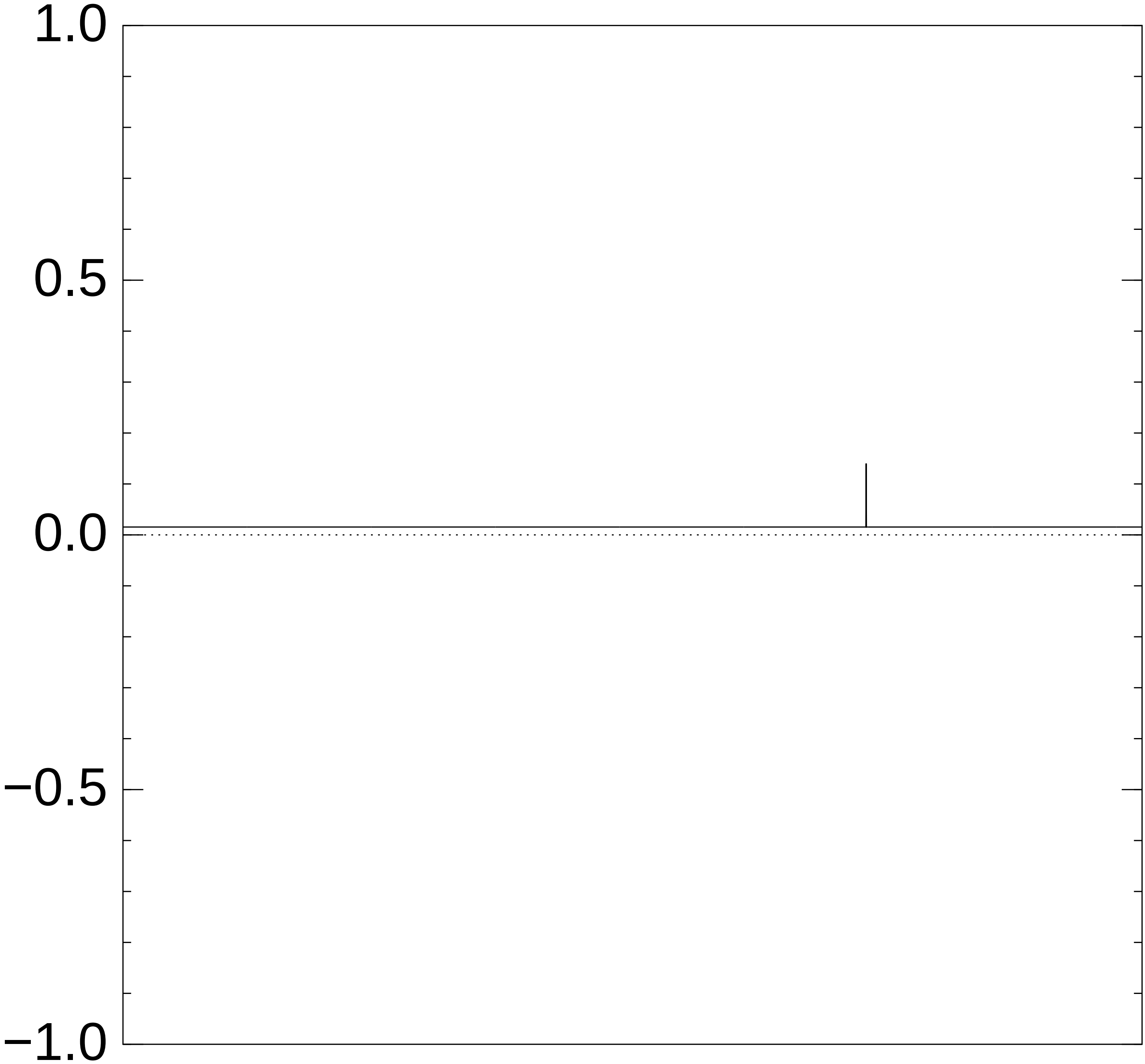
Normalized graph of $q \mapsto a_q$
for an example with $n = 12$
after $2 \times (\text{Step } 1 + \text{Step } 2)$:



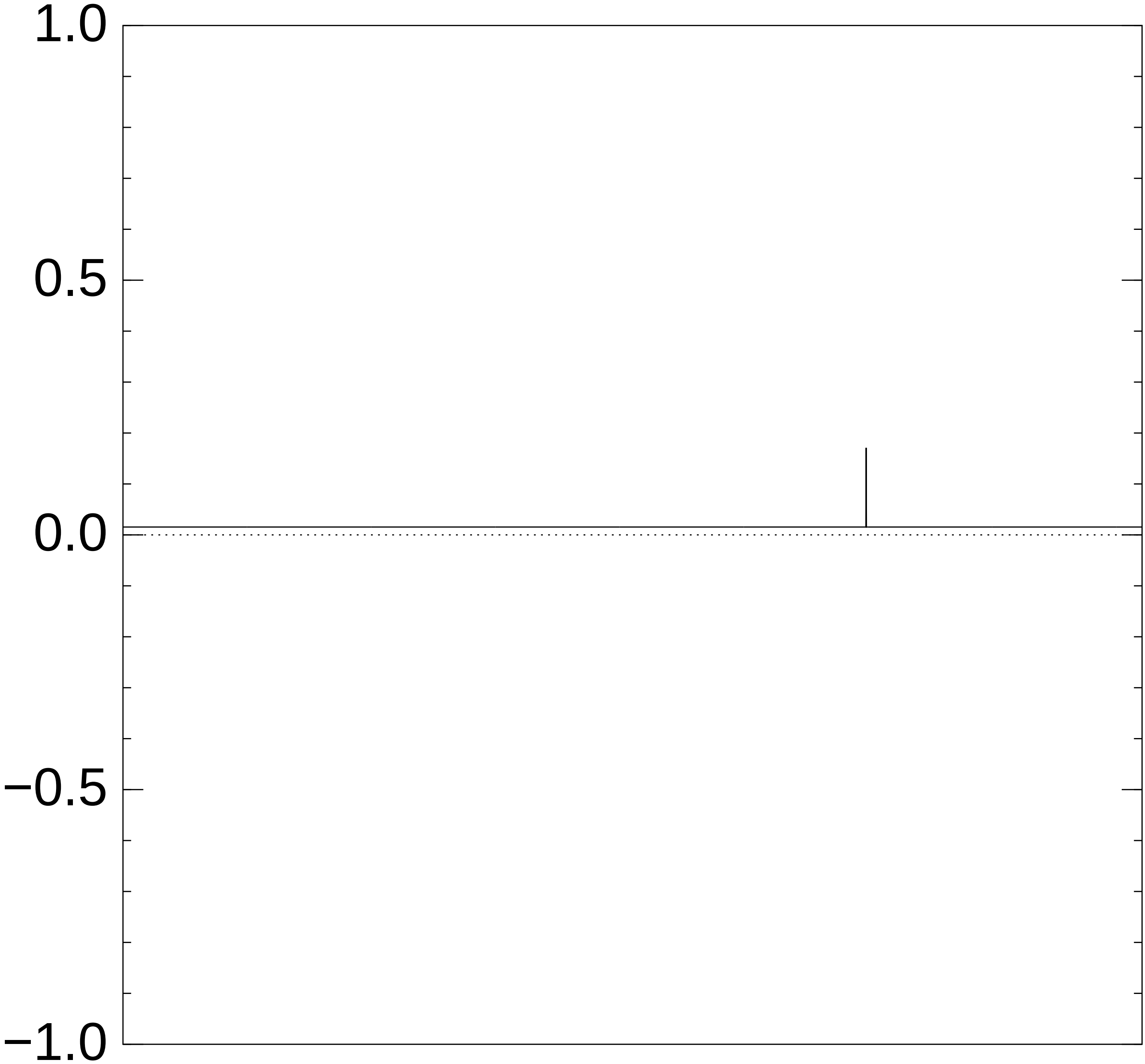
Normalized graph of $q \mapsto a_q$ for an example with $n = 12$ after $3 \times (\text{Step } 1 + \text{Step } 2)$:



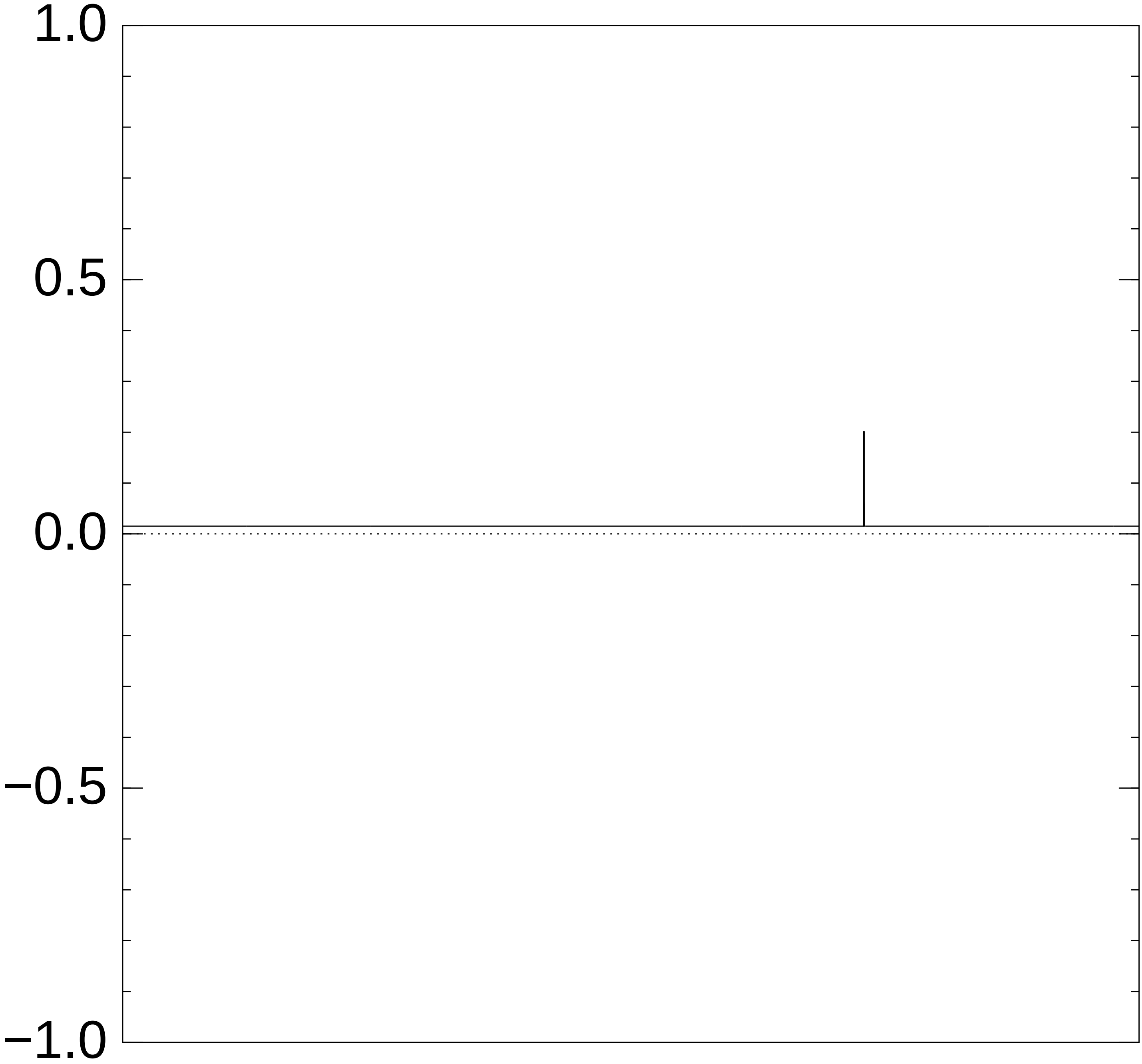
Normalized graph of $q \mapsto a_q$
for an example with $n = 12$
after $4 \times (\text{Step } 1 + \text{Step } 2)$:



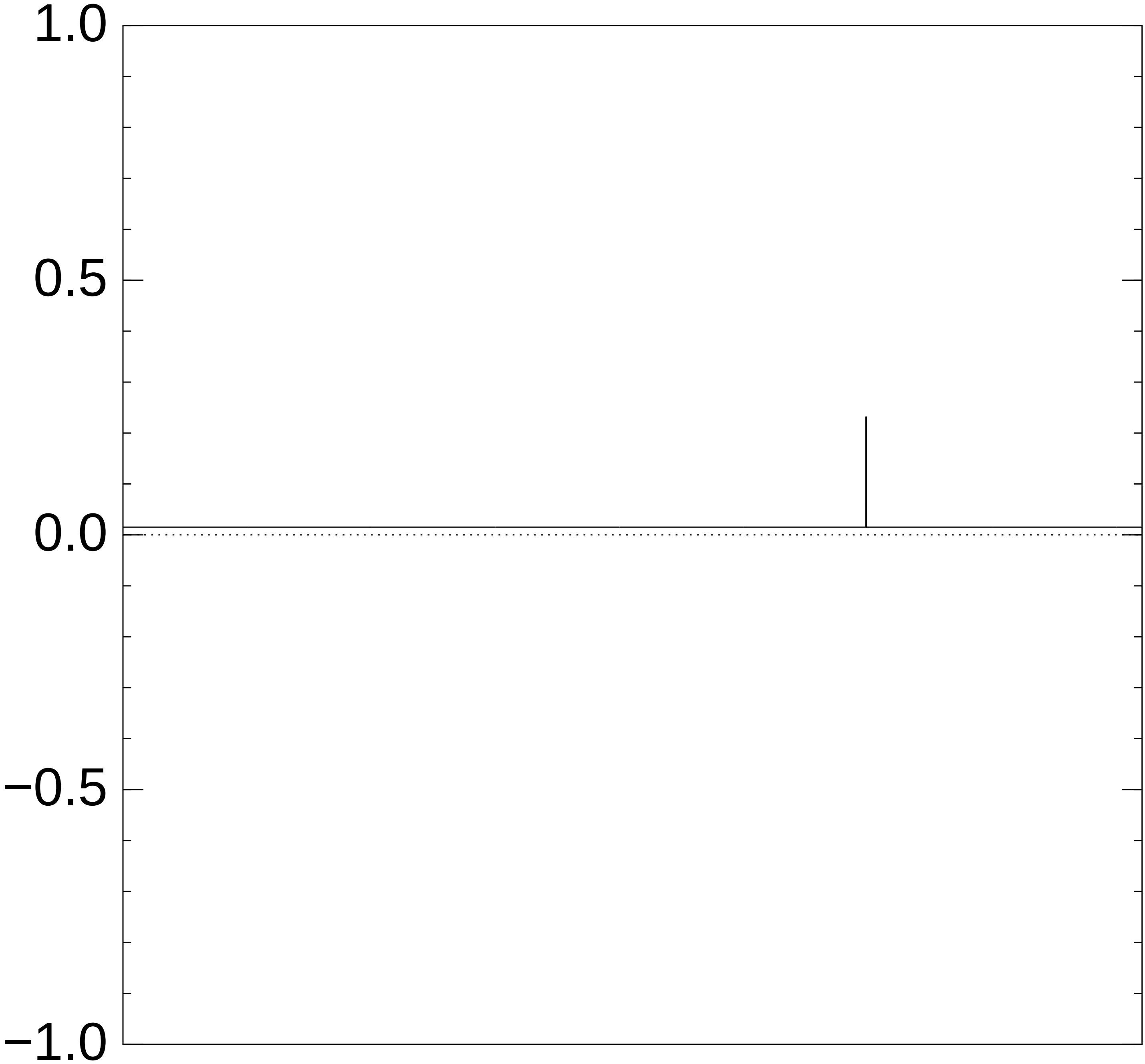
Normalized graph of $q \mapsto a_q$ for an example with $n = 12$ after $5 \times (\text{Step } 1 + \text{Step } 2)$:



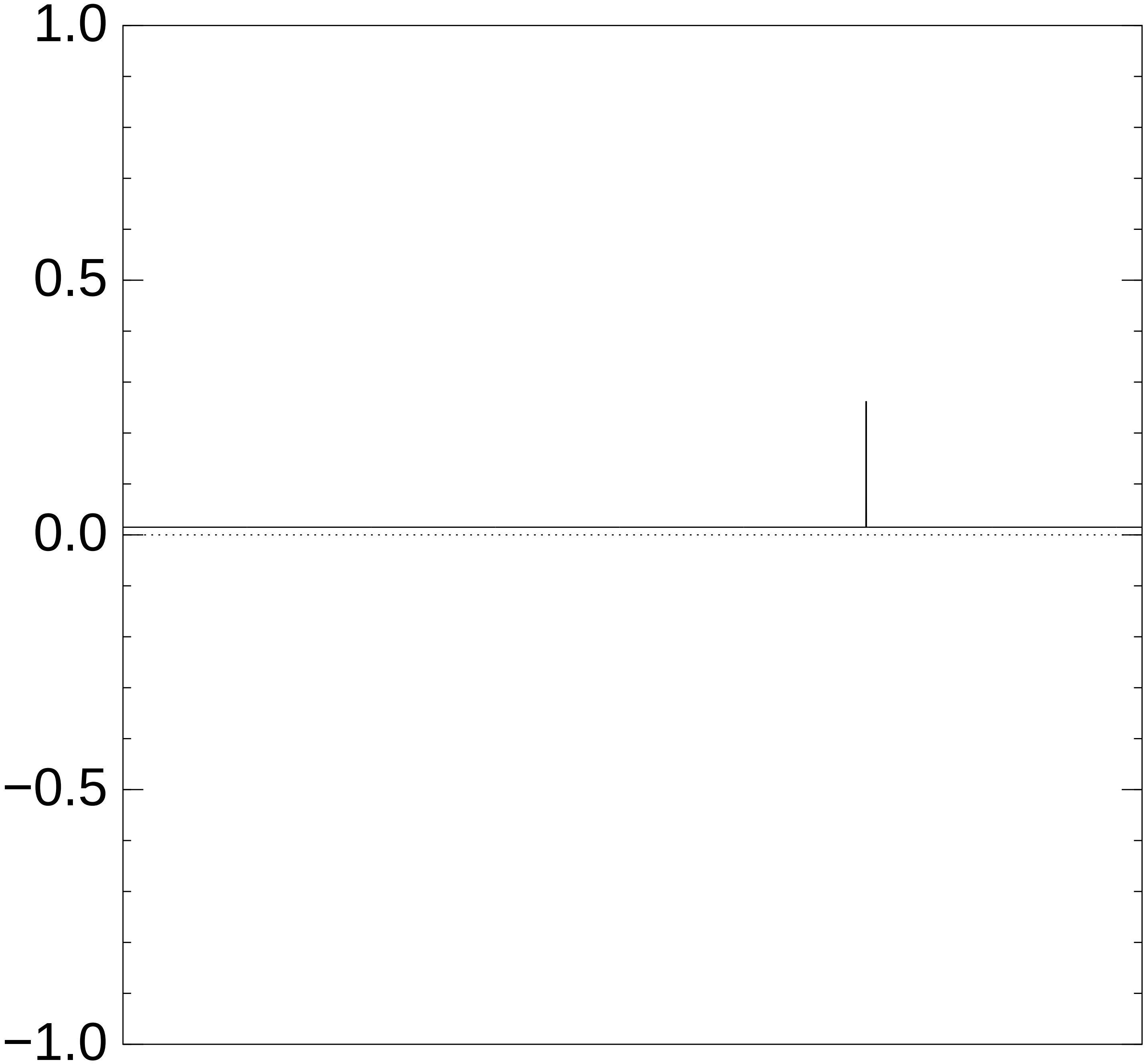
Normalized graph of $q \mapsto a_q$ for an example with $n = 12$ after $6 \times$ (Step $1 +$ Step 2):



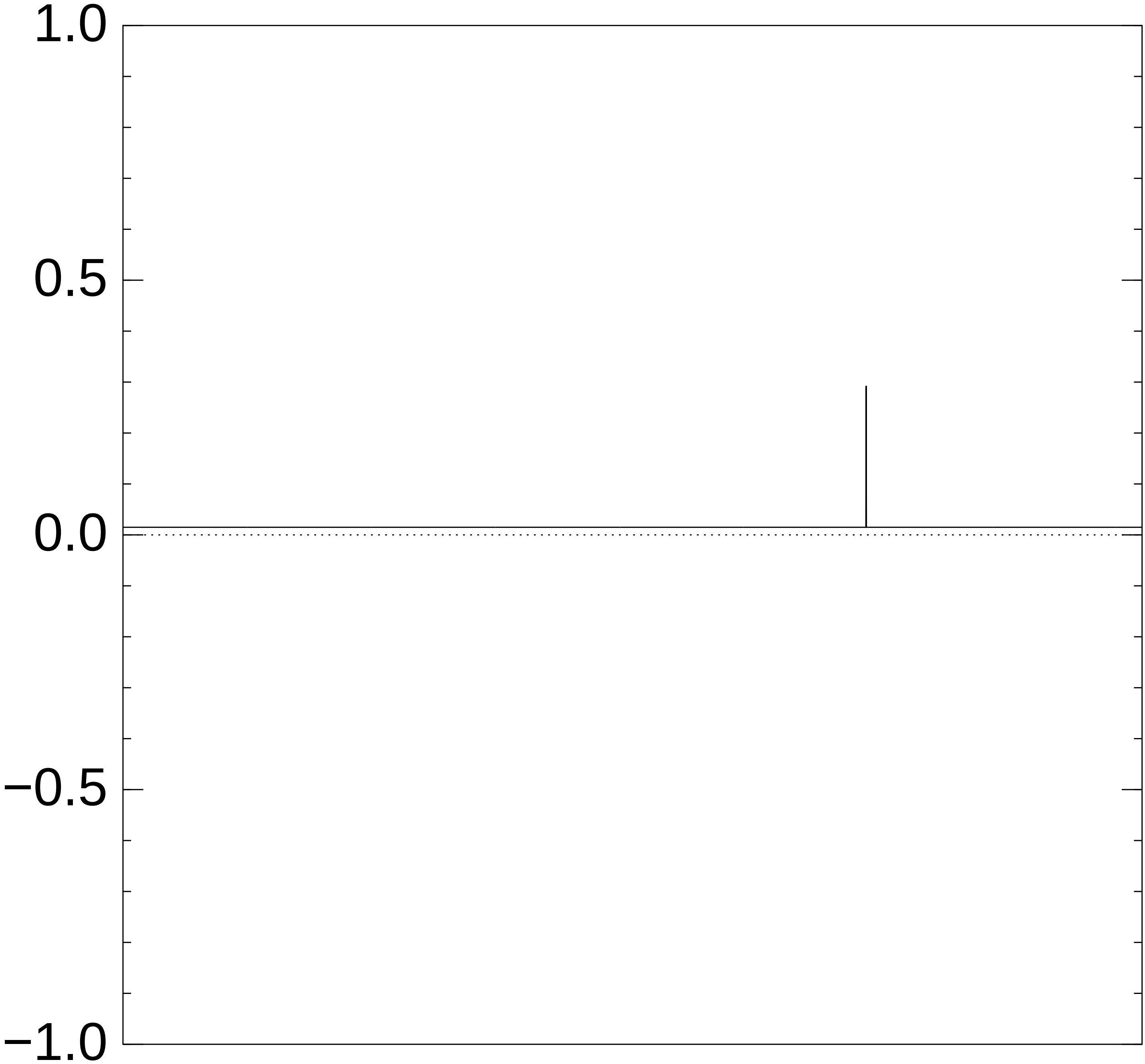
Normalized graph of $q \mapsto a_q$
for an example with $n = 12$
after $7 \times (\text{Step } 1 + \text{Step } 2)$:



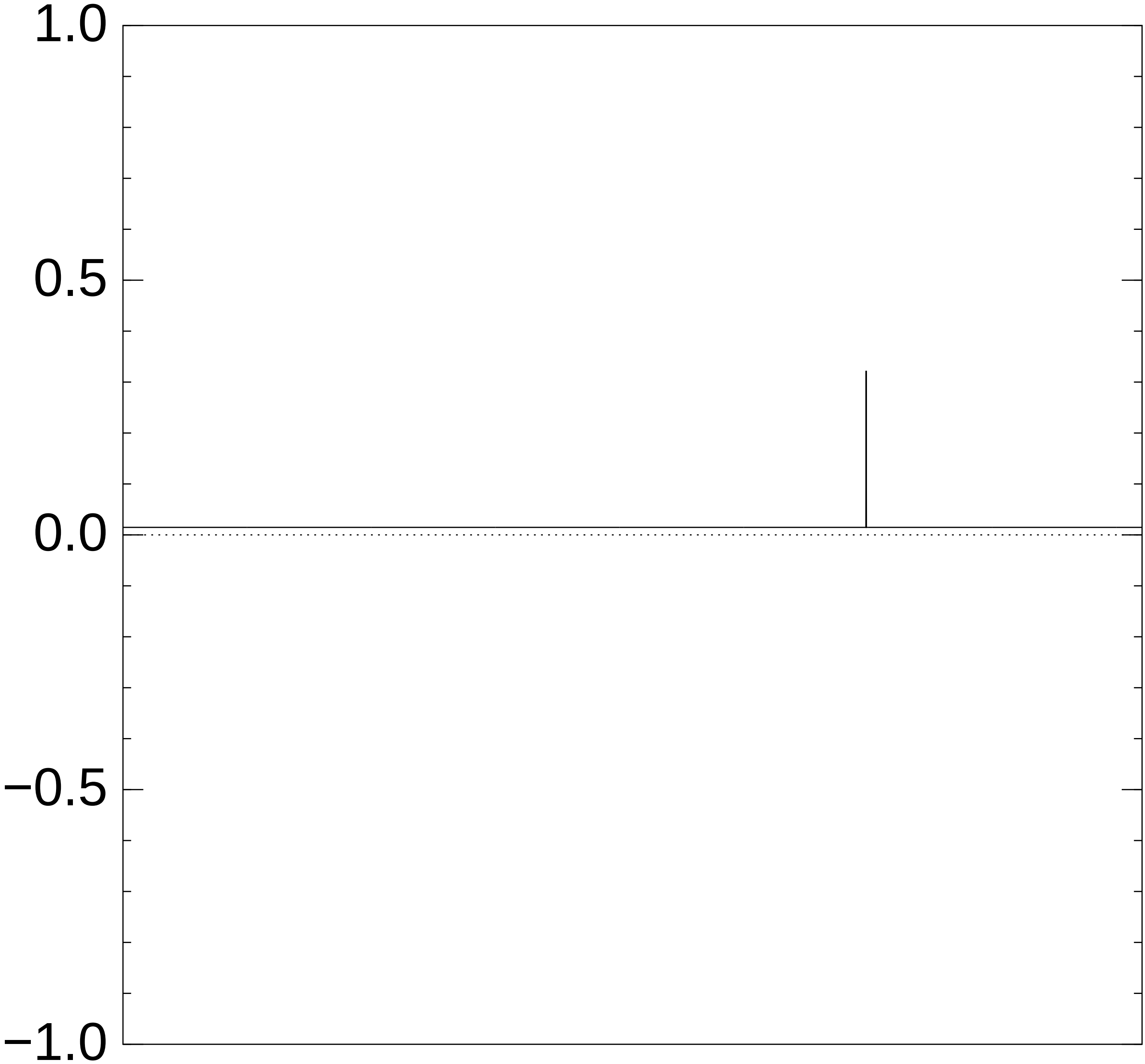
Normalized graph of $q \mapsto a_q$
for an example with $n = 12$
after $8 \times (\text{Step } 1 + \text{Step } 2)$:



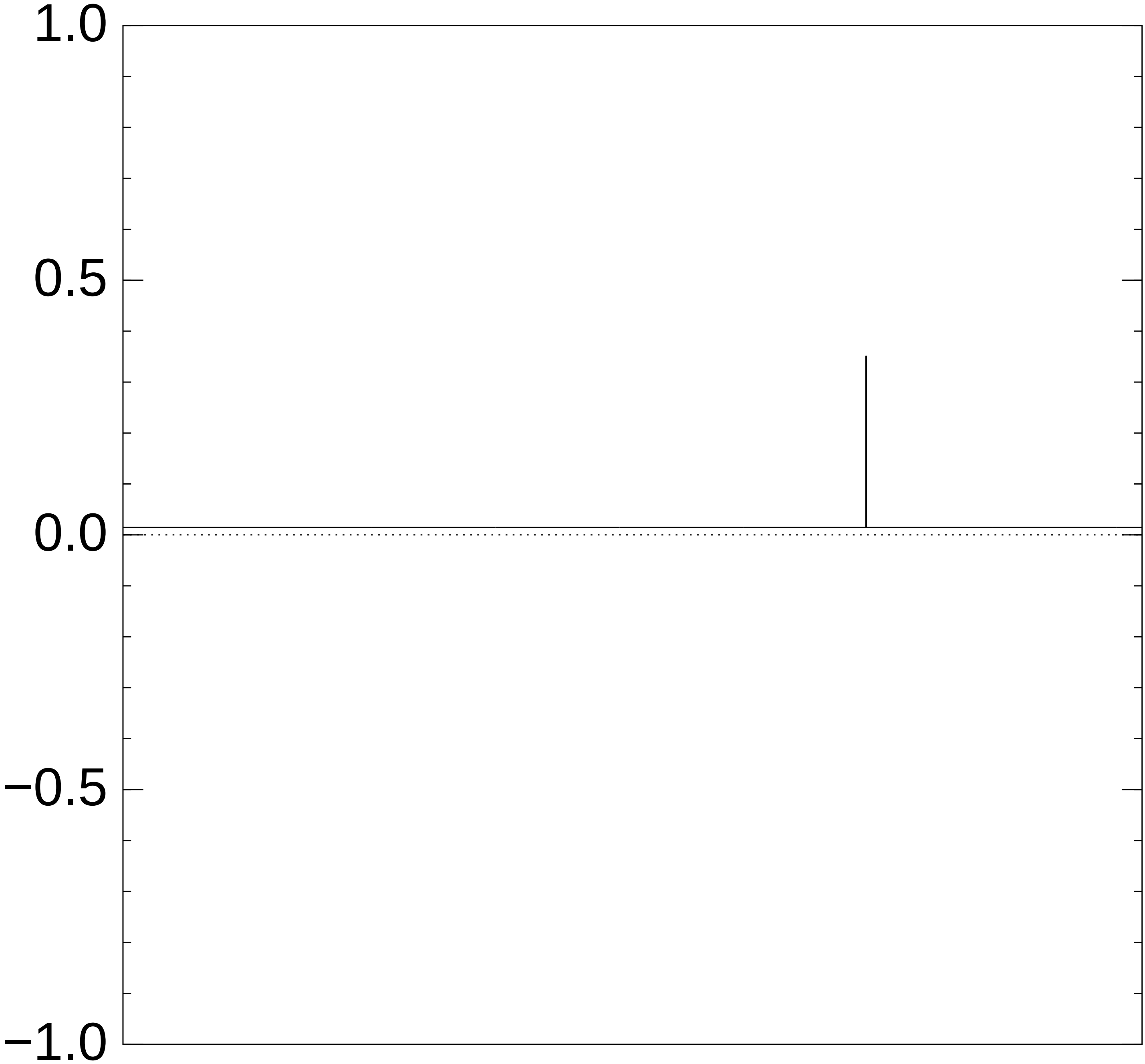
Normalized graph of $q \mapsto a_q$
for an example with $n = 12$
after $9 \times$ (Step $1 +$ Step 2):



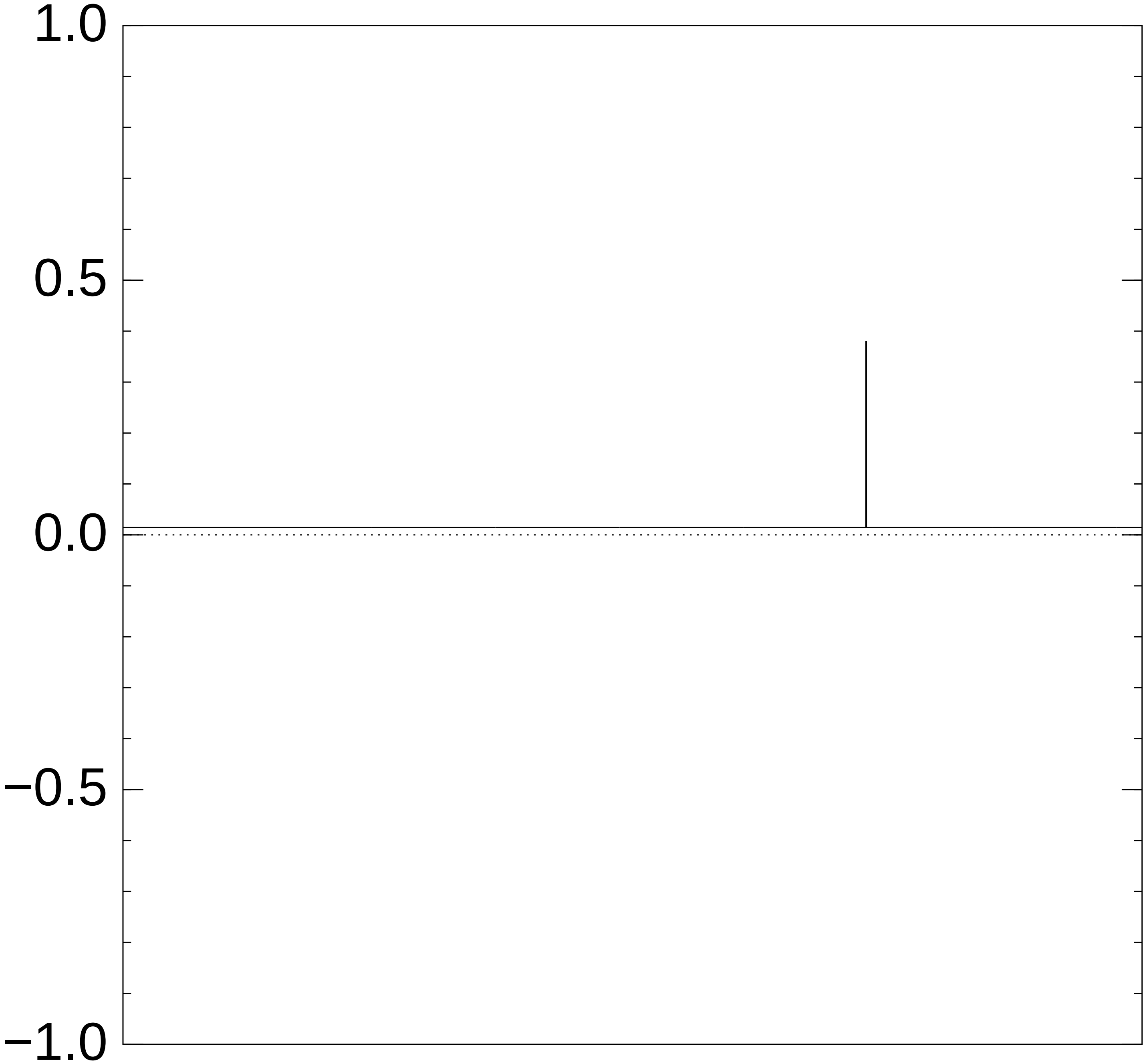
Normalized graph of $q \mapsto a_q$
for an example with $n = 12$
after $10 \times (\text{Step } 1 + \text{Step } 2)$:



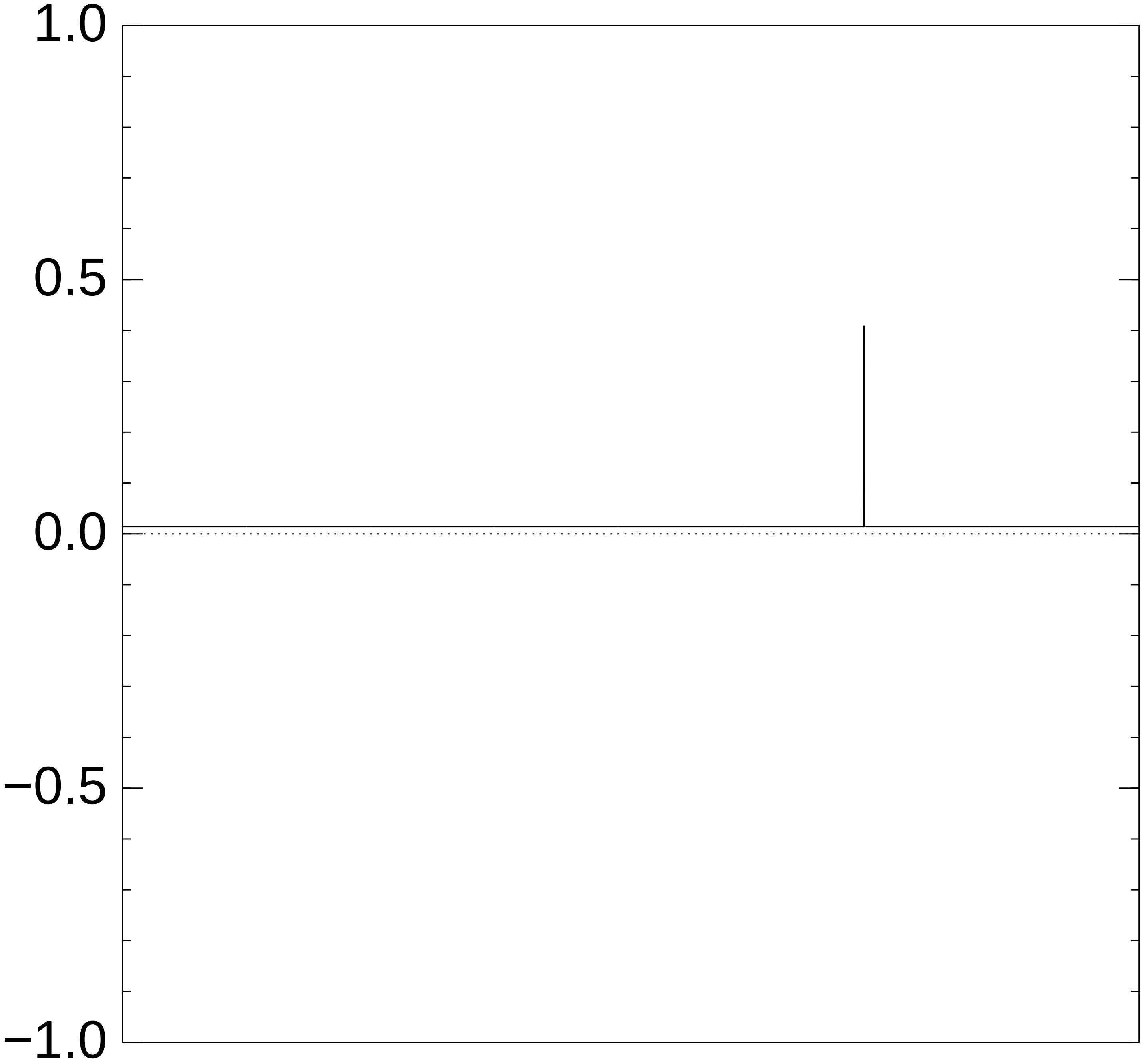
Normalized graph of $q \mapsto a_q$
for an example with $n = 12$
after $11 \times$ (Step $1 +$ Step 2):



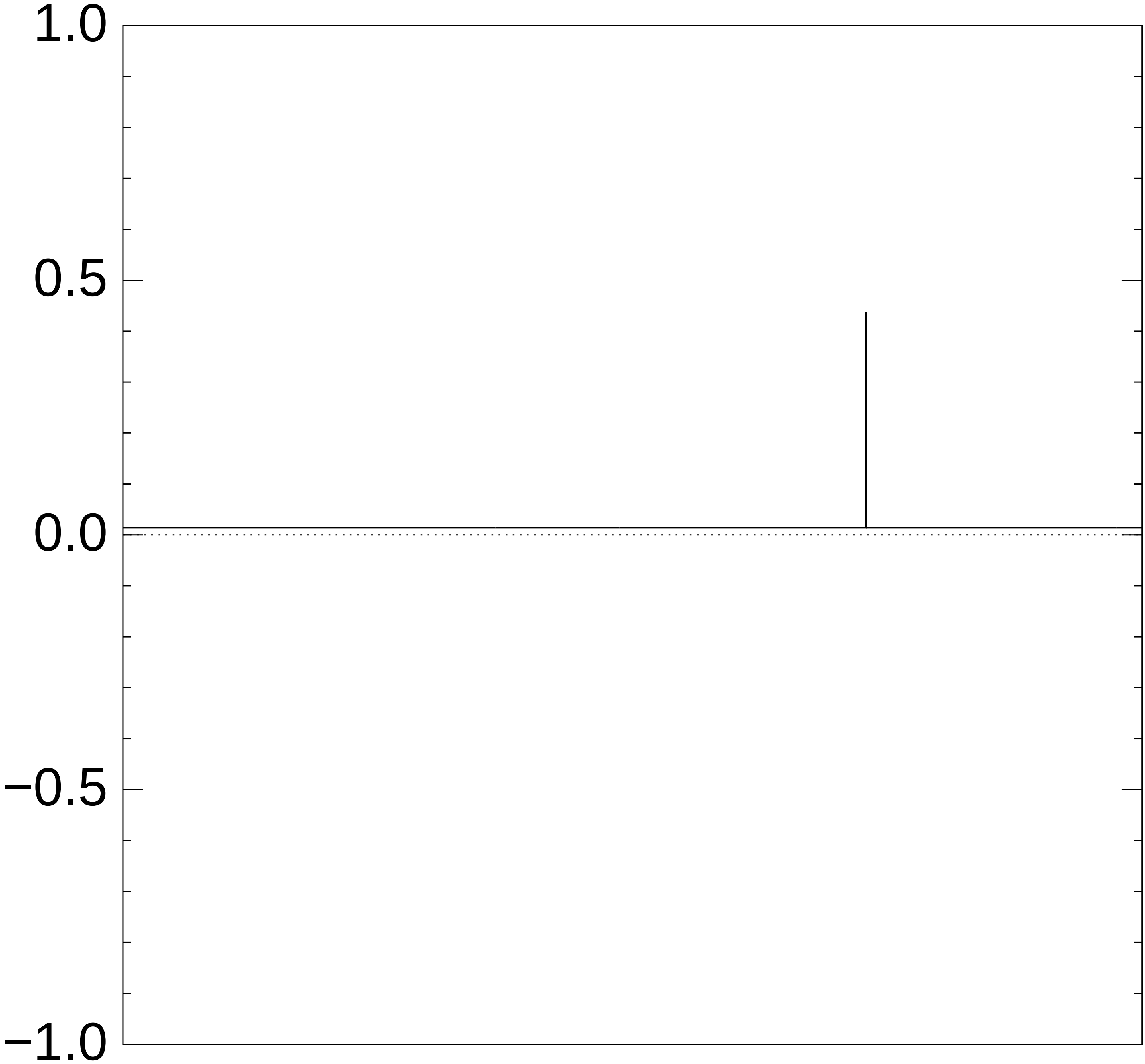
Normalized graph of $q \mapsto a_q$
for an example with $n = 12$
after $12 \times (\text{Step } 1 + \text{Step } 2)$:



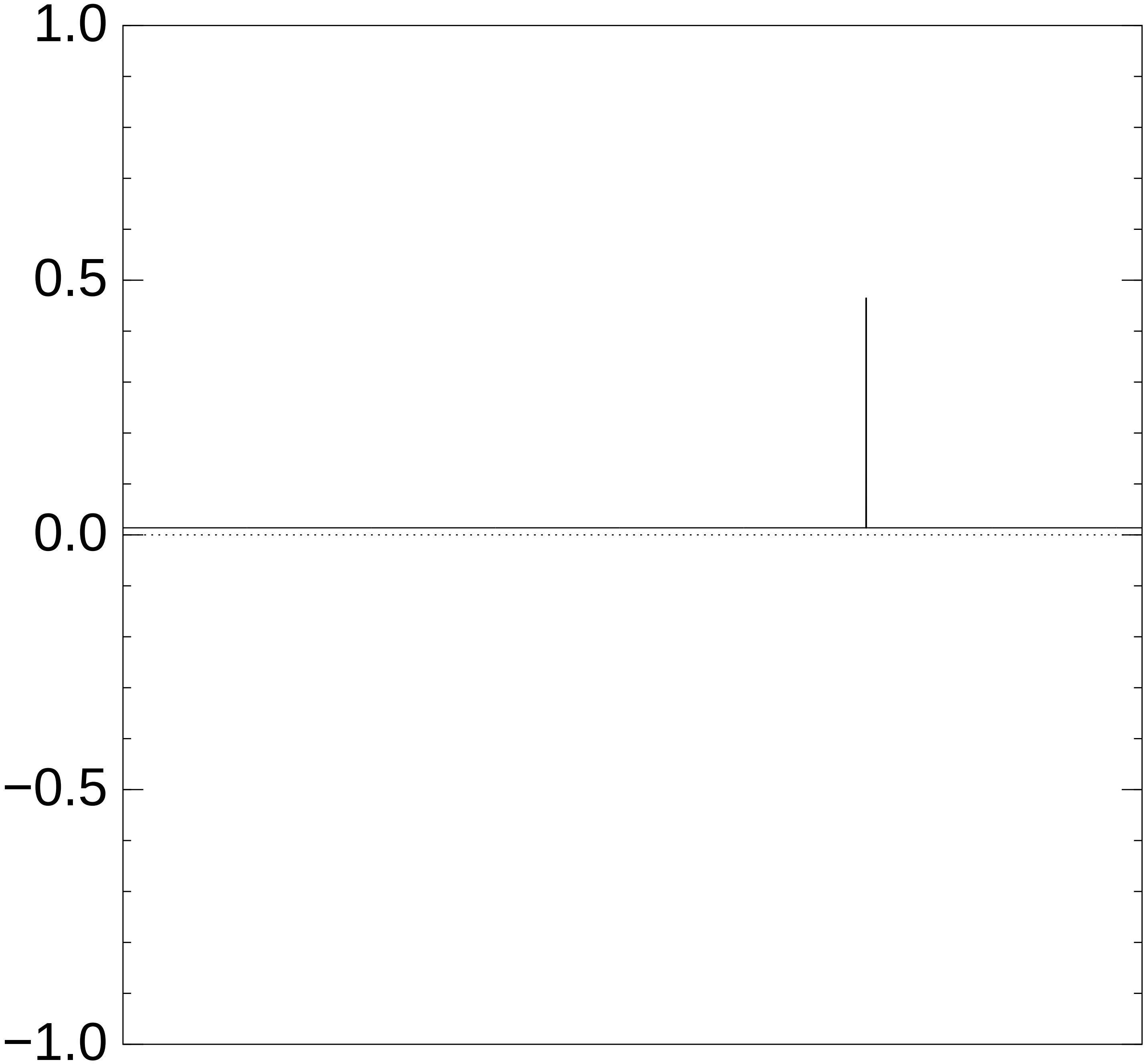
Normalized graph of $q \mapsto a_q$ for an example with $n = 12$ after $13 \times$ (Step 1 + Step 2):



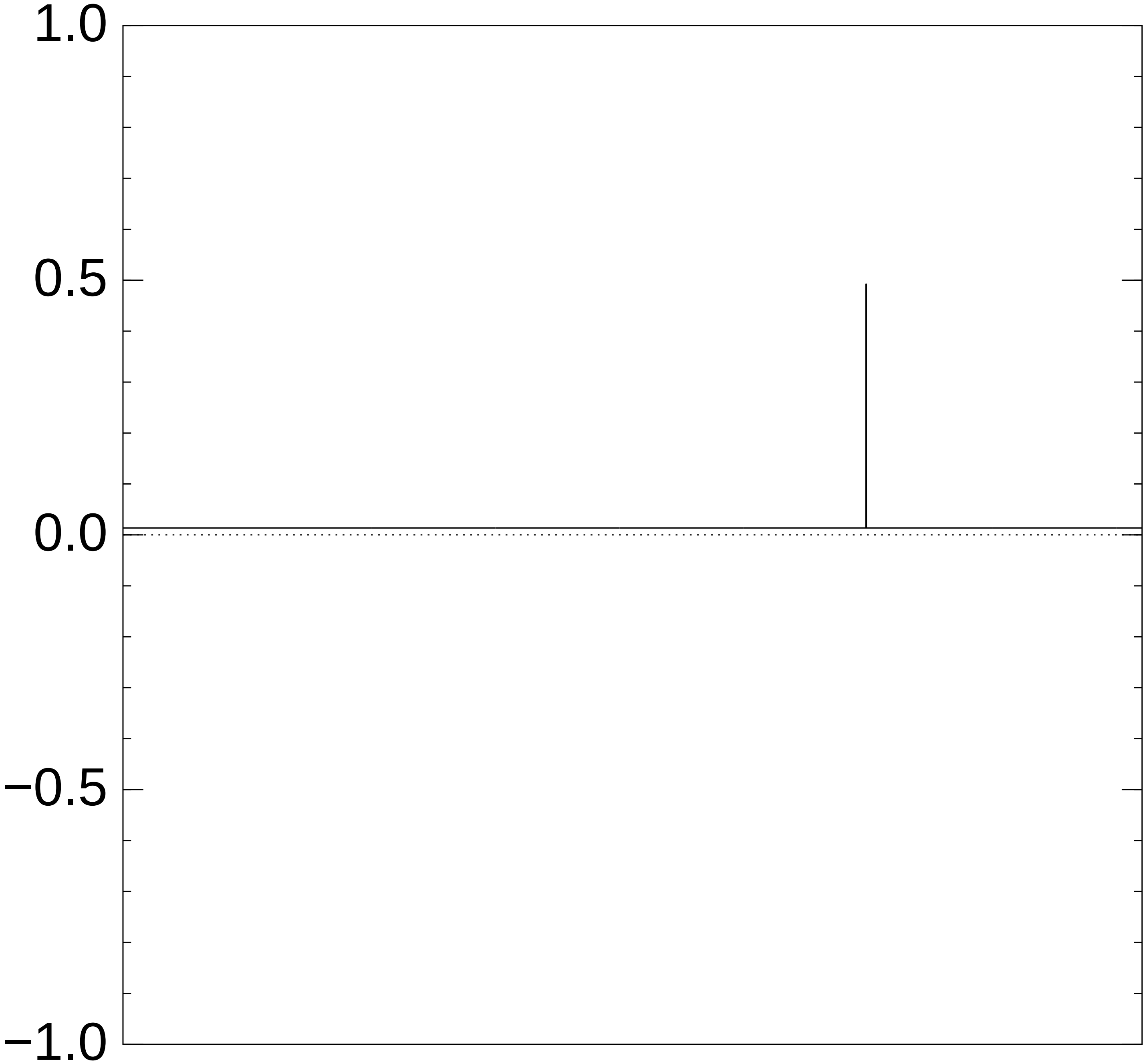
Normalized graph of $q \mapsto a_q$
for an example with $n = 12$
after $14 \times$ (Step $1 +$ Step 2):



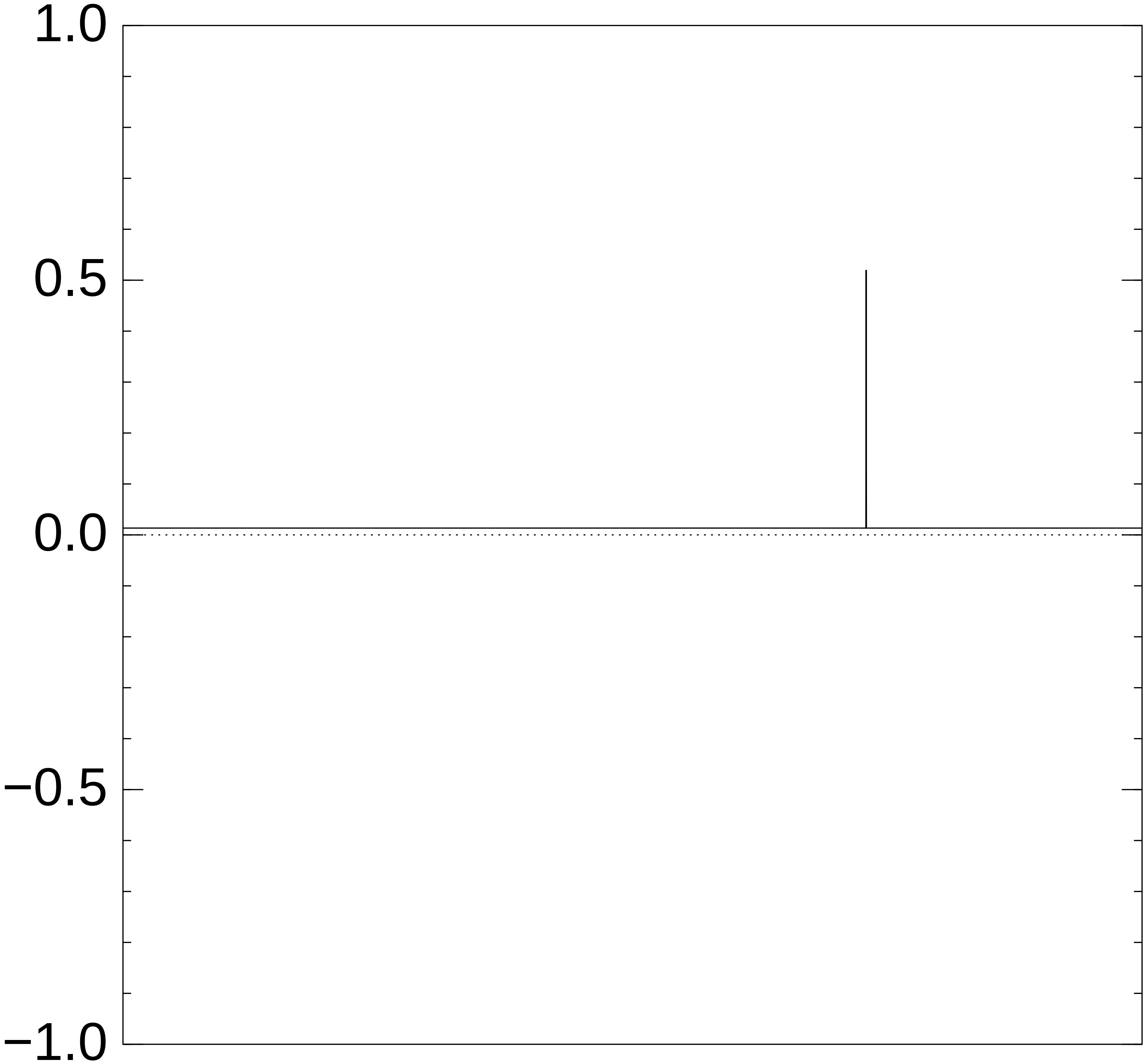
Normalized graph of $q \mapsto a_q$ for an example with $n = 12$ after $15 \times$ (Step $1 +$ Step 2):



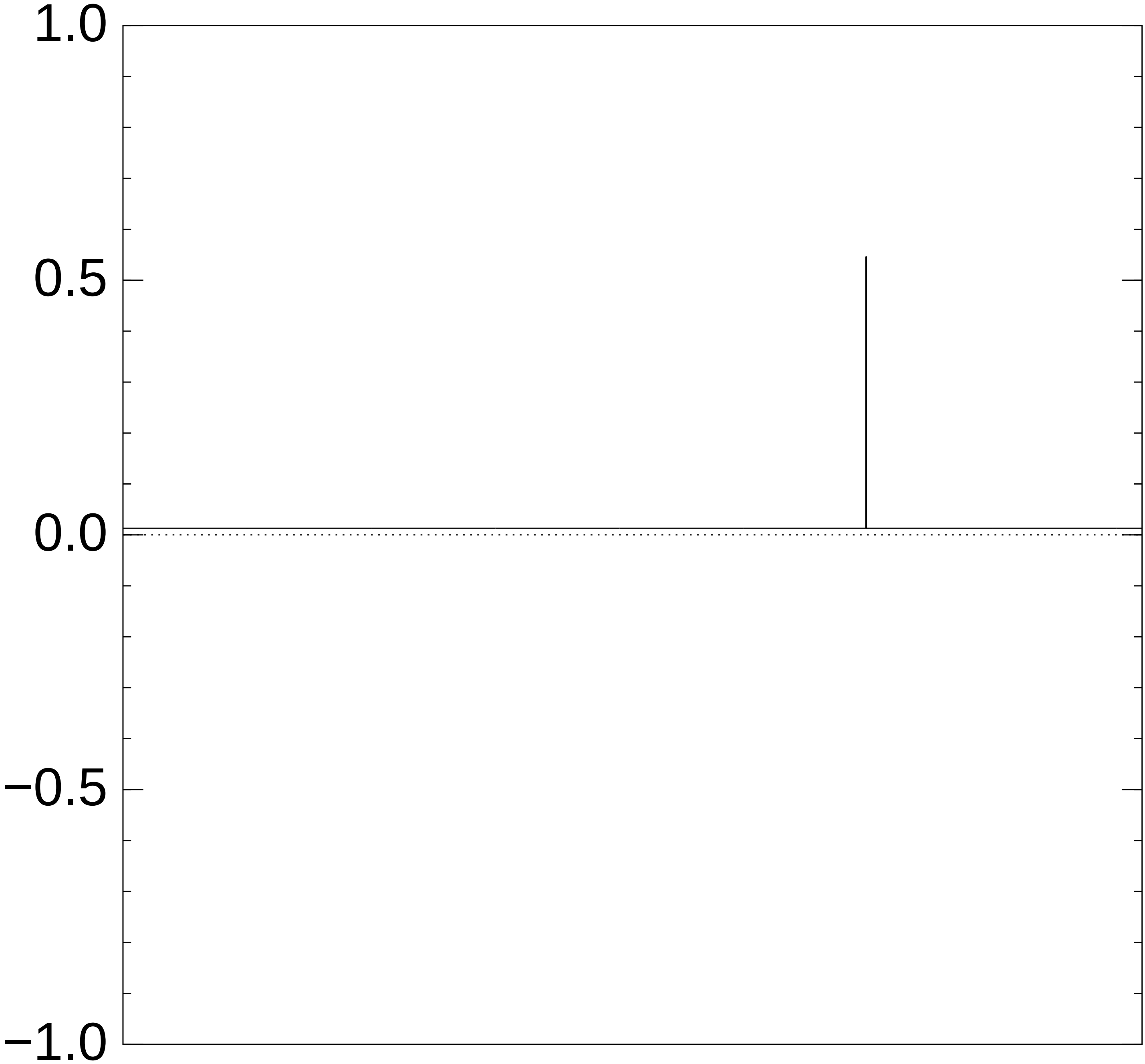
Normalized graph of $q \mapsto a_q$ for an example with $n = 12$ after $16 \times$ (Step $1 +$ Step 2):



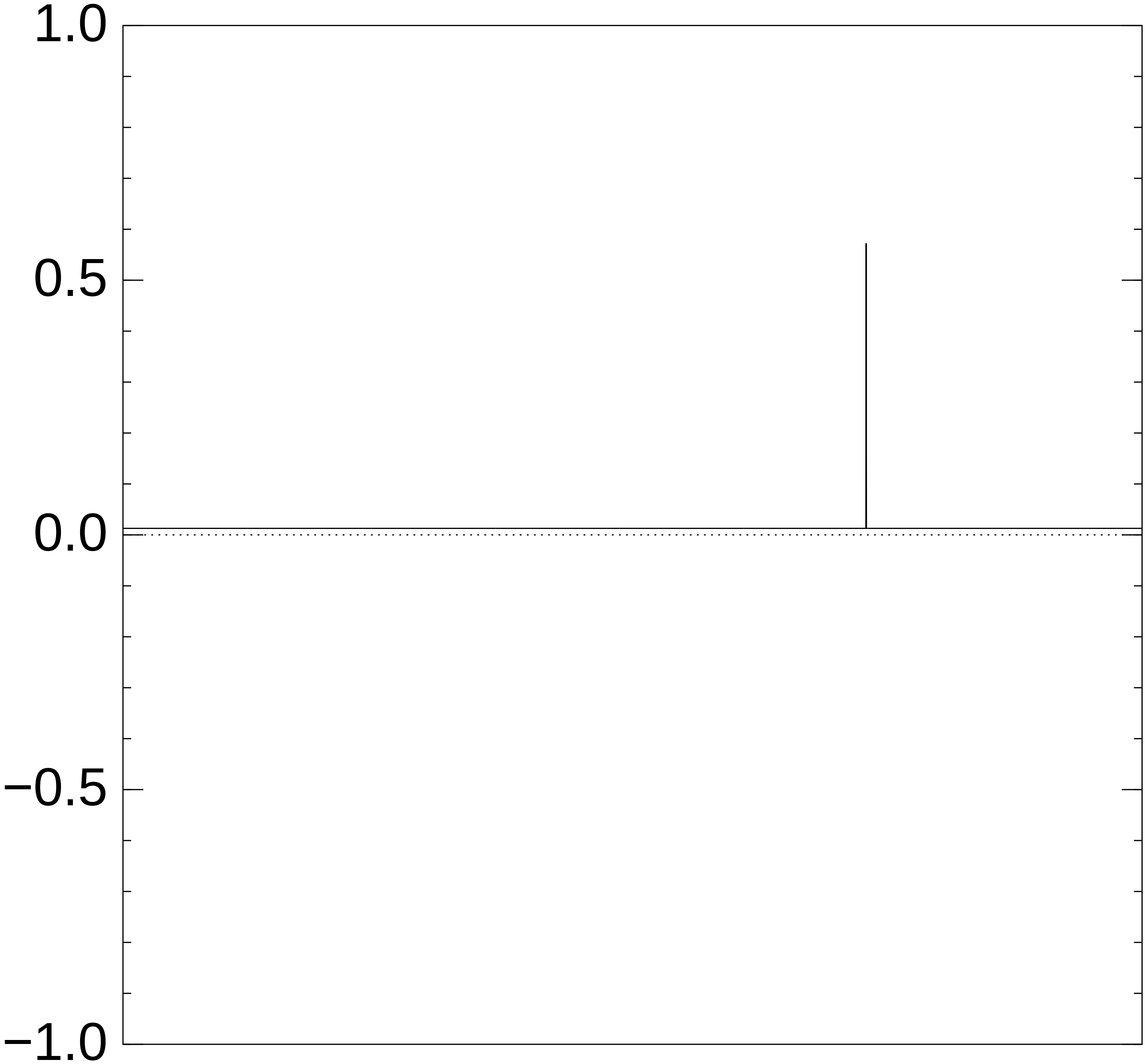
Normalized graph of $q \mapsto a_q$ for an example with $n = 12$ after $17 \times$ (Step $1 +$ Step 2):



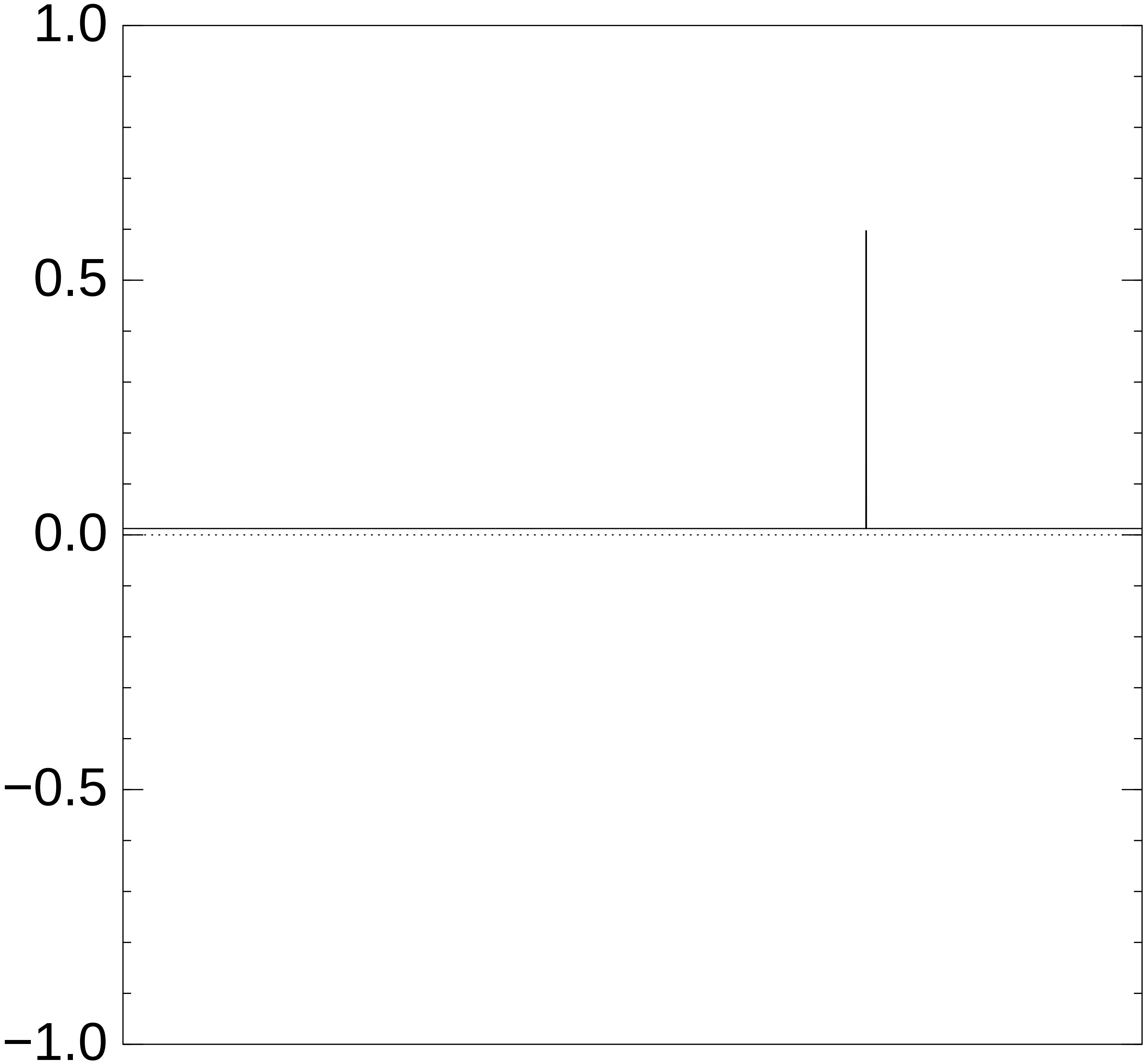
Normalized graph of $q \mapsto a_q$
for an example with $n = 12$
after $18 \times$ (Step 1 + Step 2):



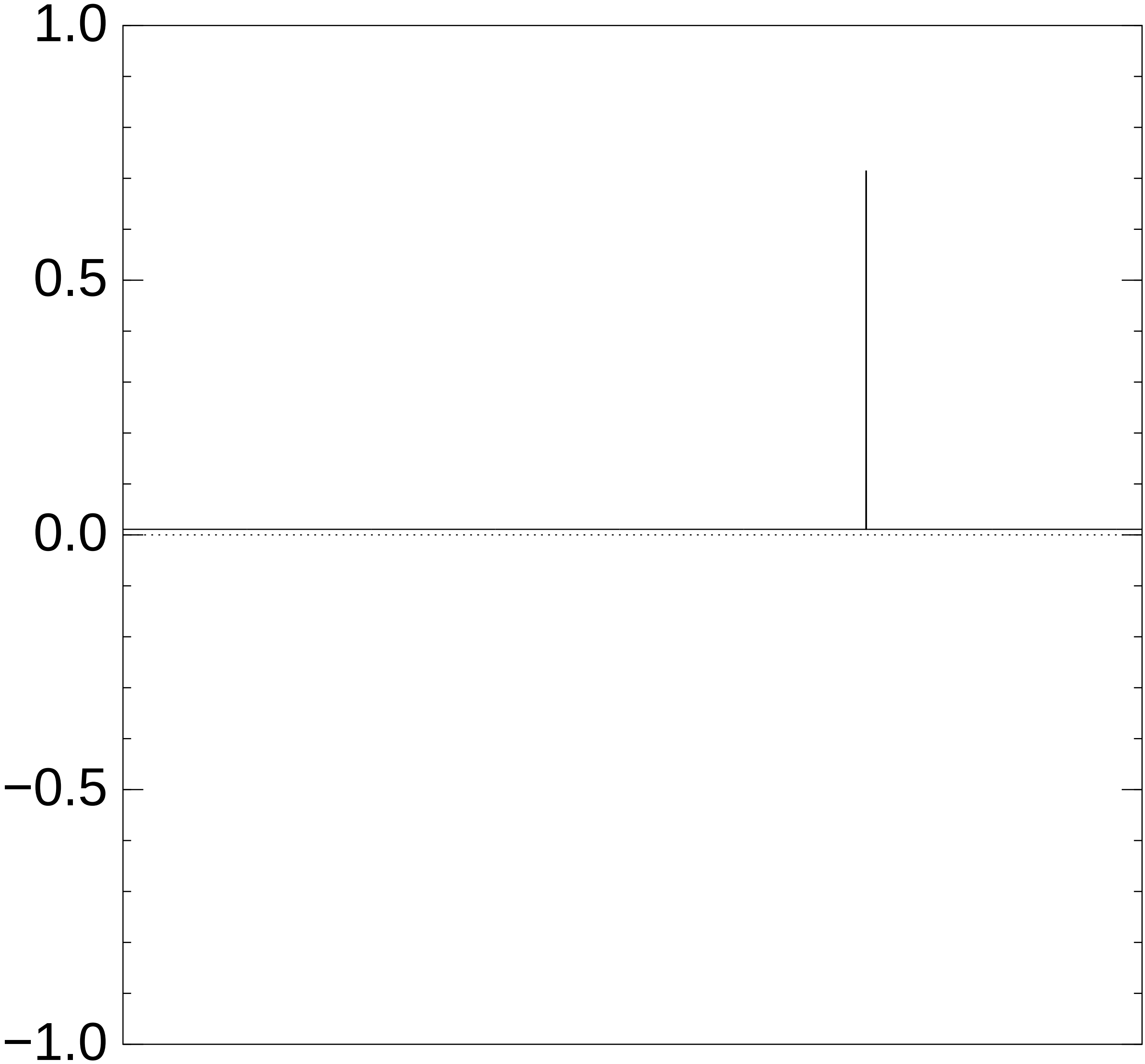
Normalized graph of $q \mapsto a_q$ for an example with $n = 12$ after $19 \times$ (Step $1 +$ Step 2):



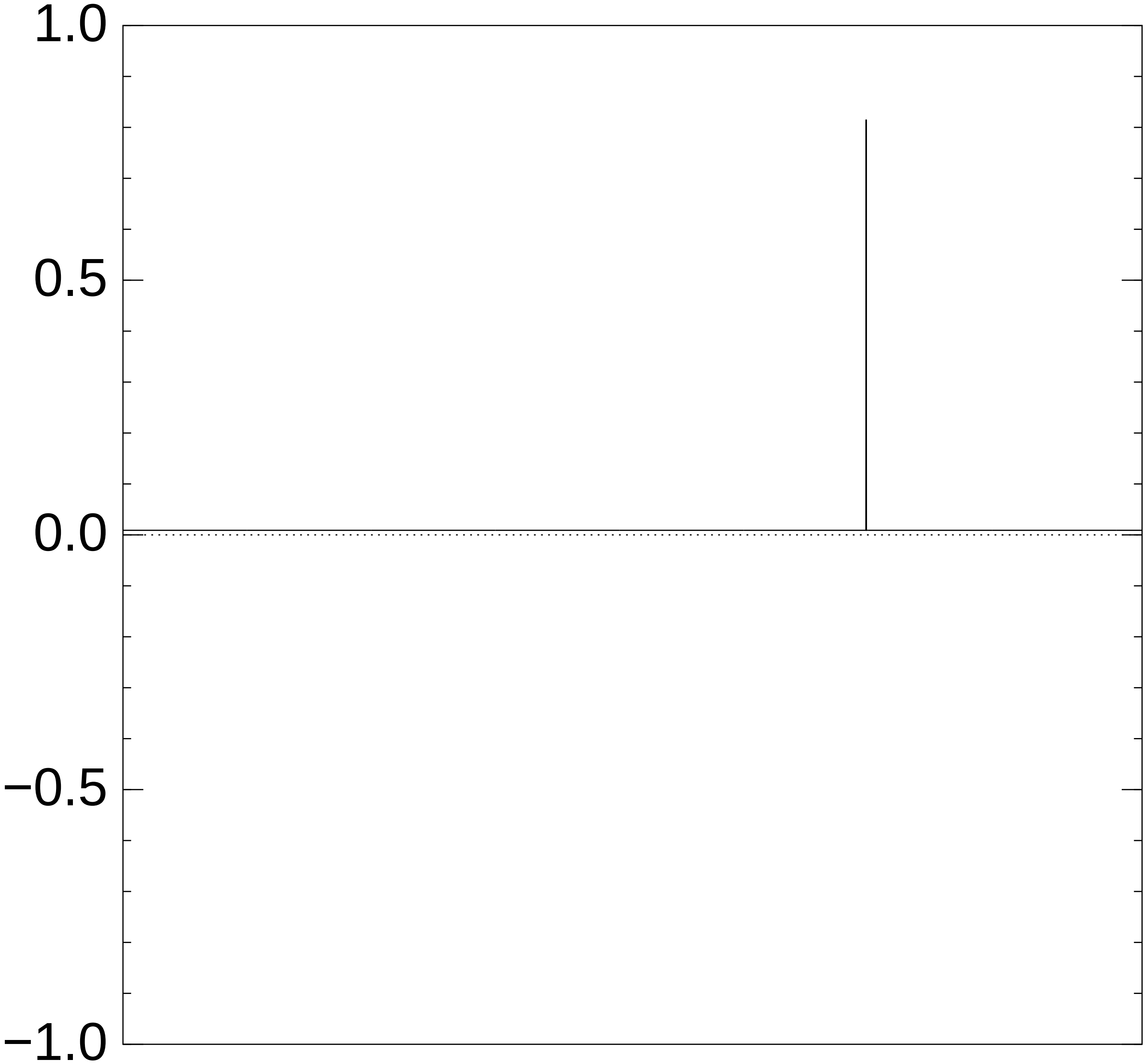
Normalized graph of $q \mapsto a_q$ for an example with $n = 12$ after $20 \times$ (Step $1 +$ Step 2):



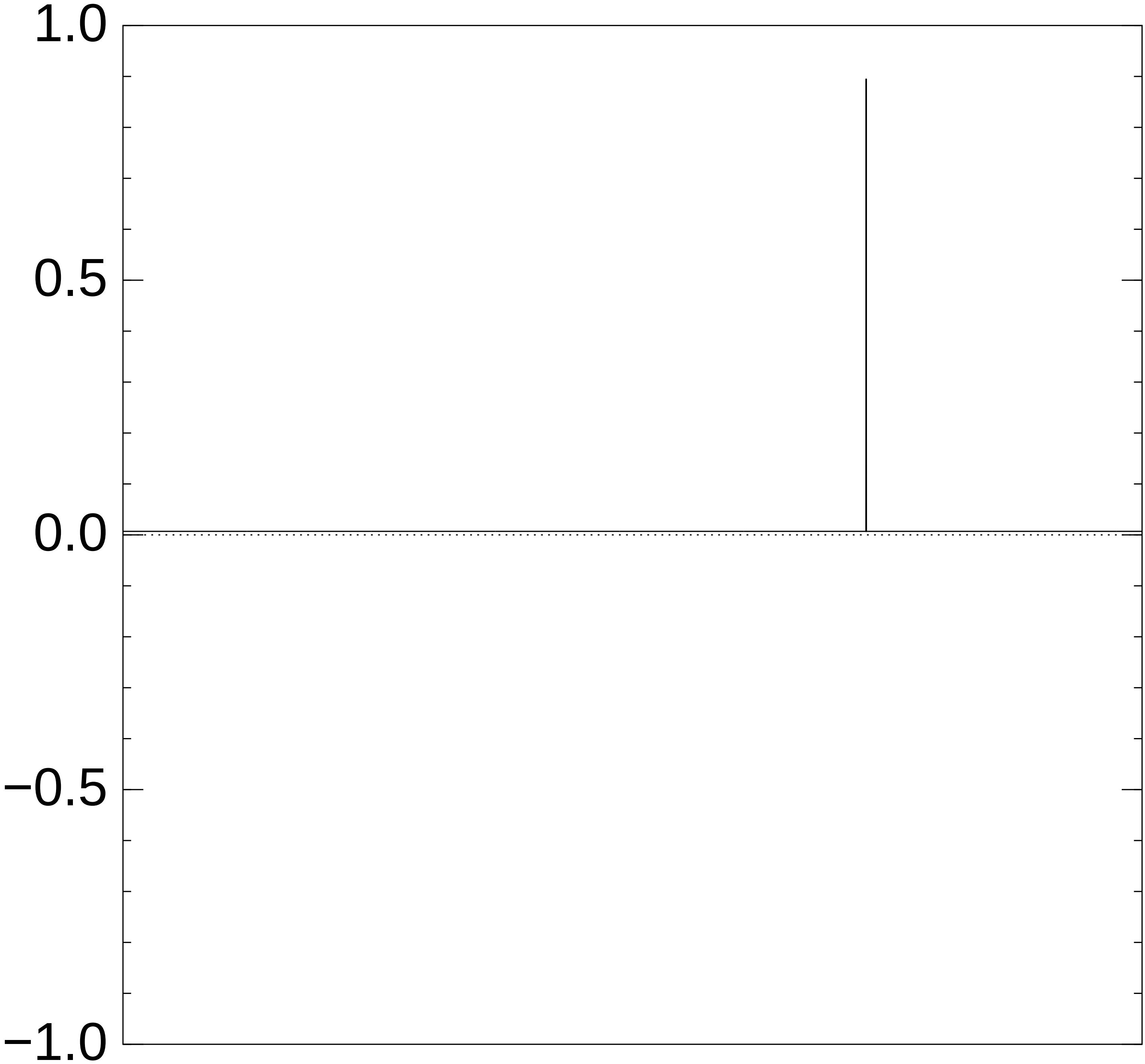
Normalized graph of $q \mapsto a_q$
for an example with $n = 12$
after $25 \times (\text{Step } 1 + \text{Step } 2)$:



Normalized graph of $q \mapsto a_q$
for an example with $n = 12$
after $30 \times (\text{Step } 1 + \text{Step } 2)$:

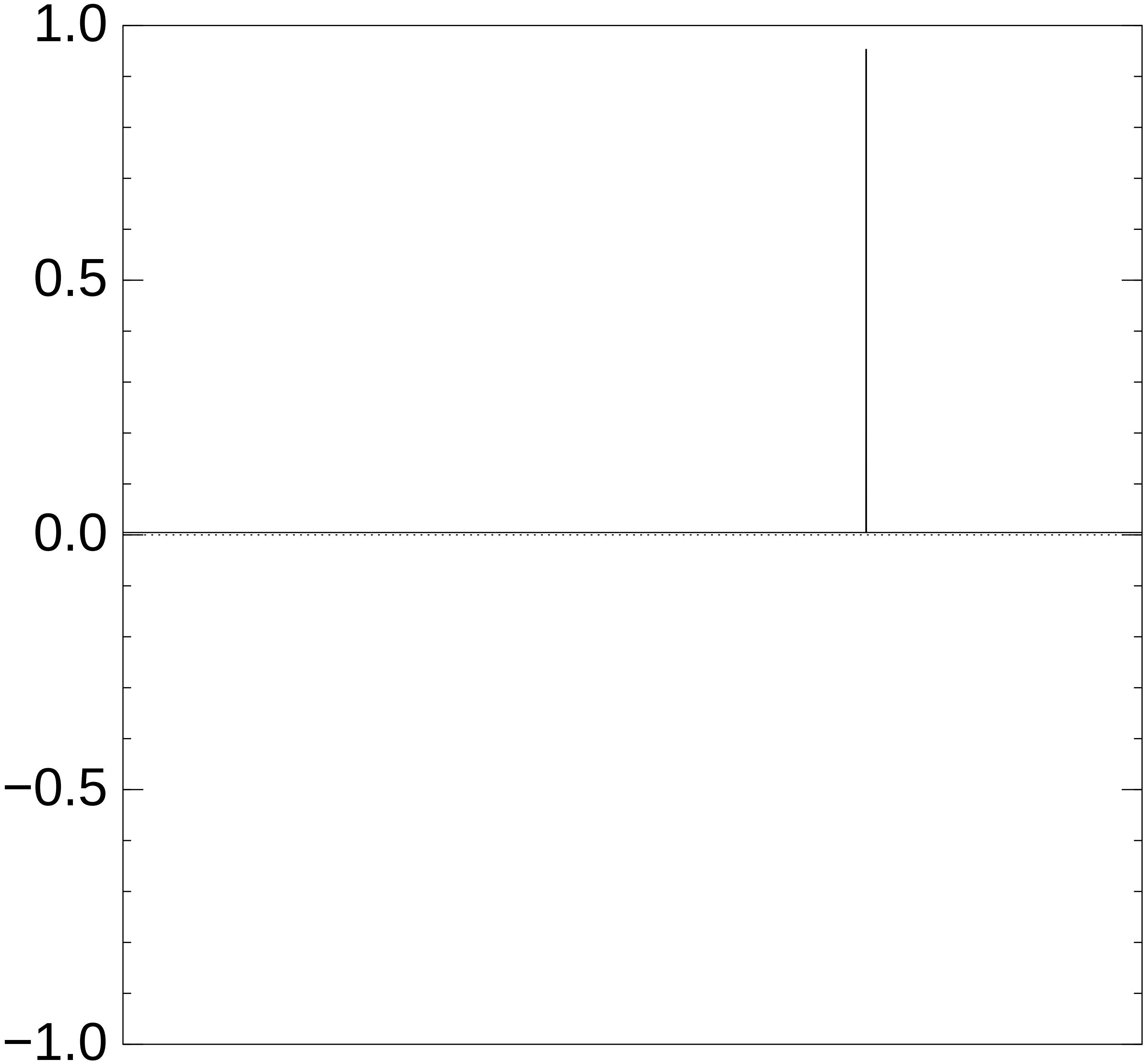


Normalized graph of $q \mapsto a_q$ for an example with $n = 12$ after $35 \times$ (Step $1 +$ Step 2):

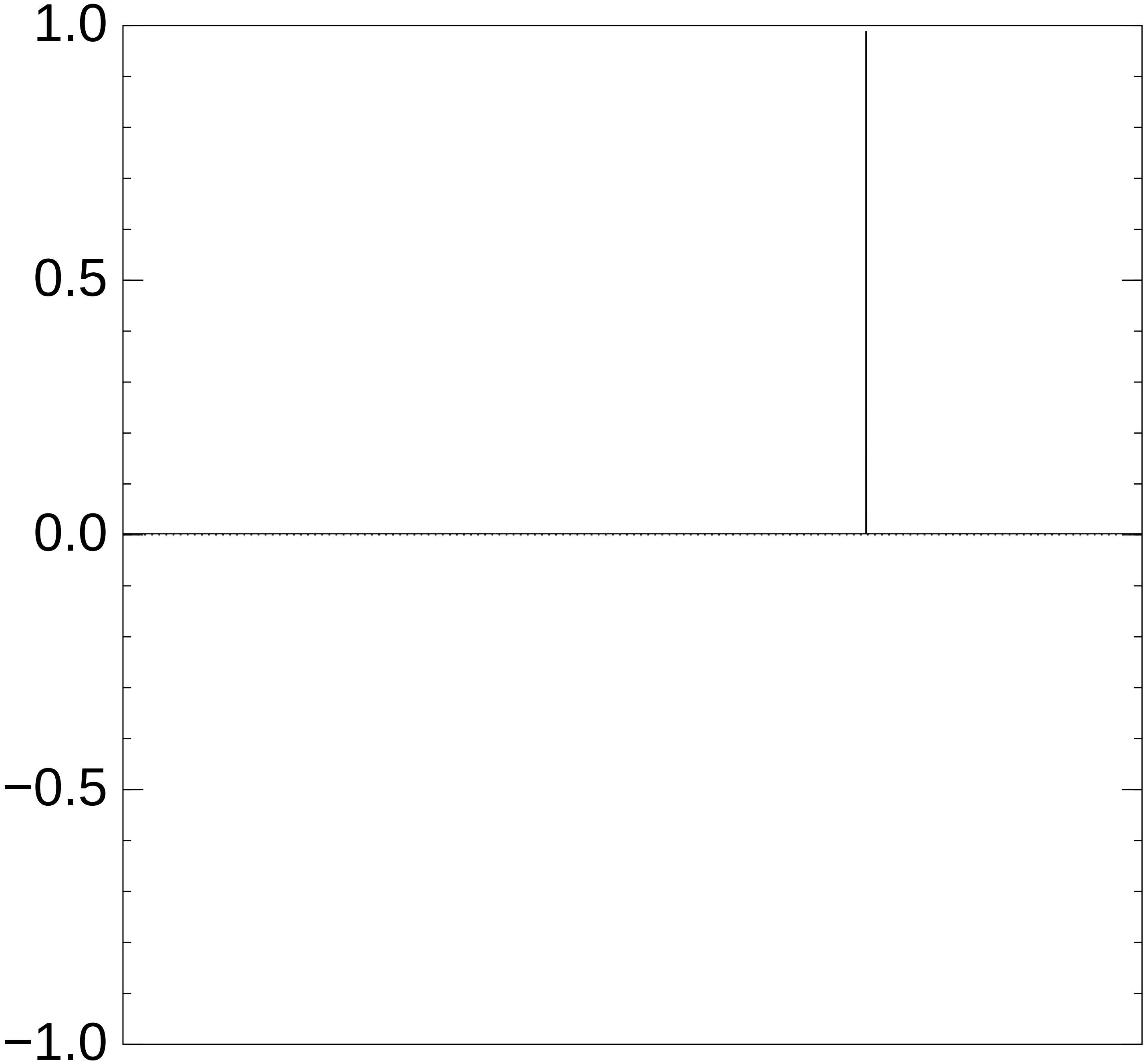


Good moment to stop, measure.

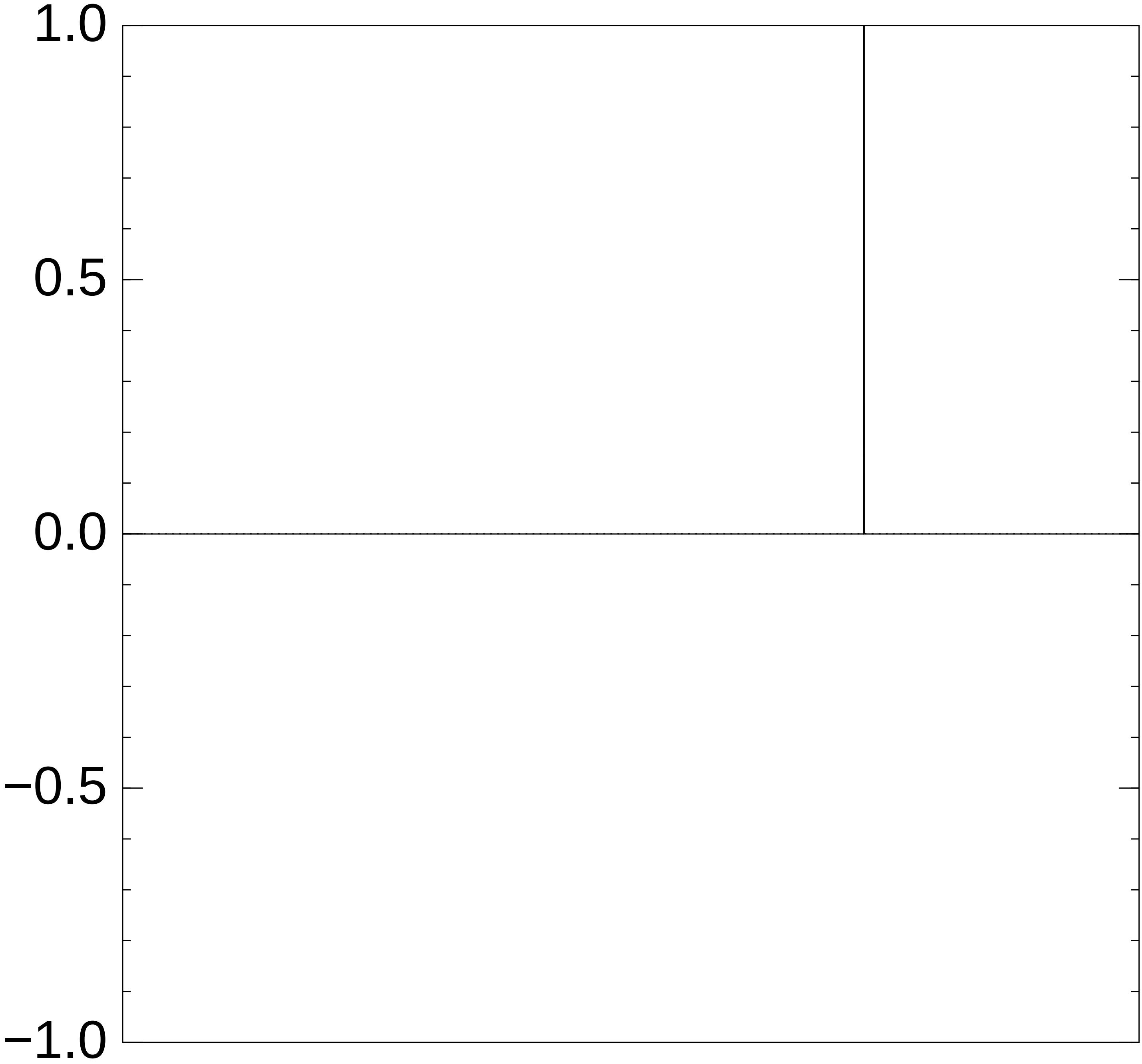
Normalized graph of $q \mapsto a_q$ for an example with $n = 12$ after $40 \times$ (Step $1 +$ Step 2):



Normalized graph of $q \mapsto a_q$ for an example with $n = 12$ after $45 \times (\text{Step 1} + \text{Step 2})$:

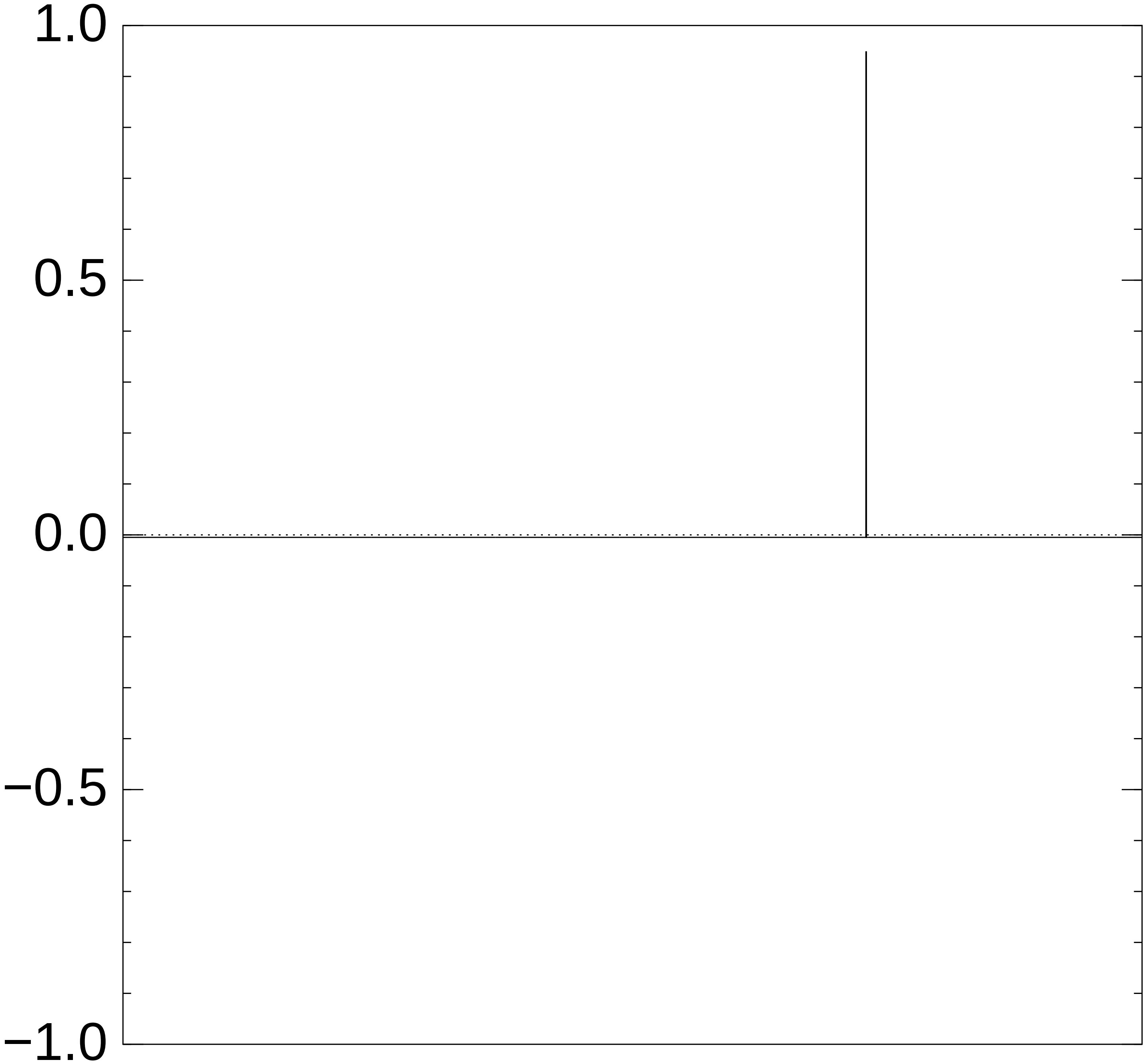


Normalized graph of $q \mapsto a_q$
for an example with $n = 12$
after $50 \times$ (Step $1 +$ Step 2):

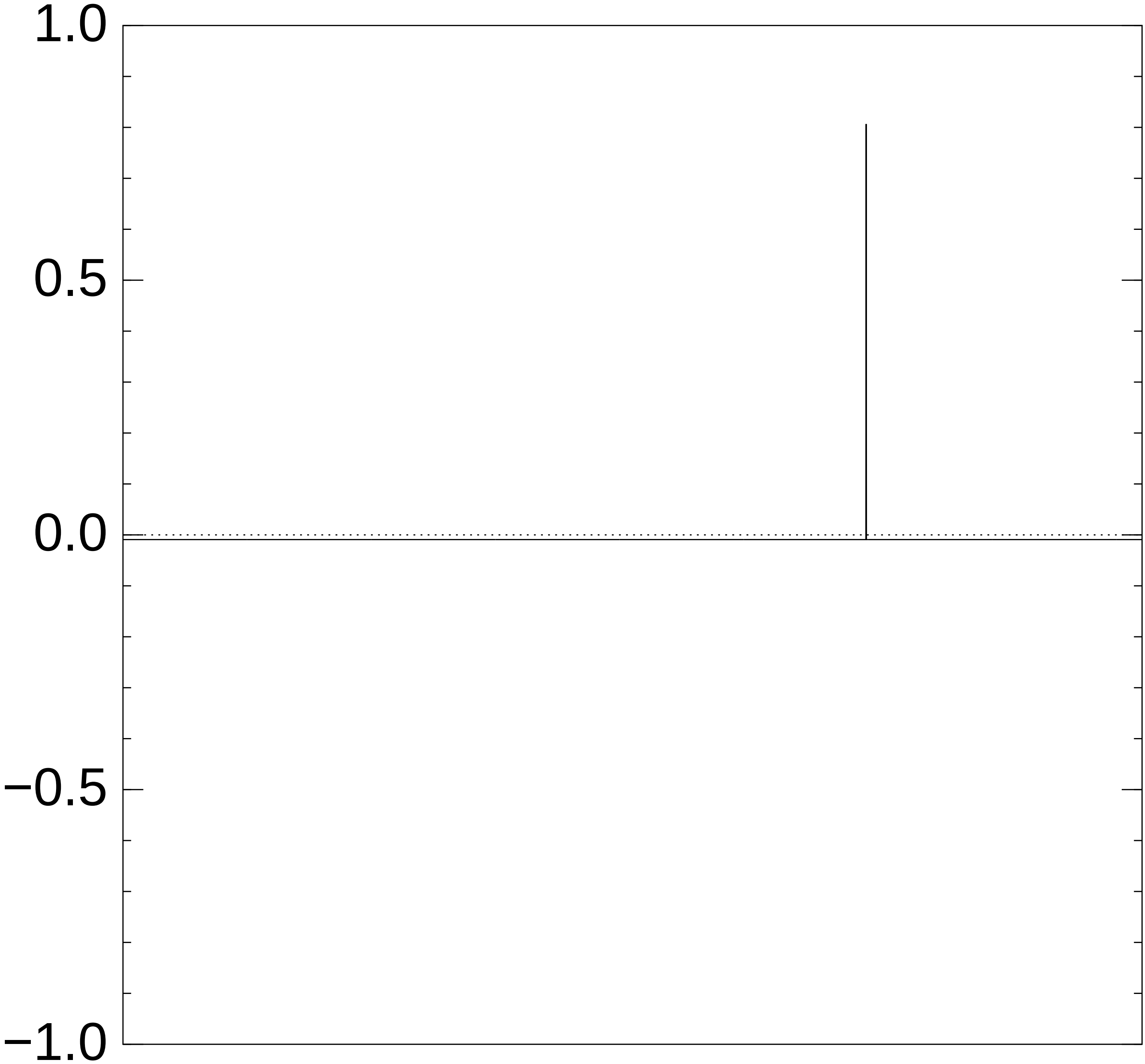


Traditional stopping point.

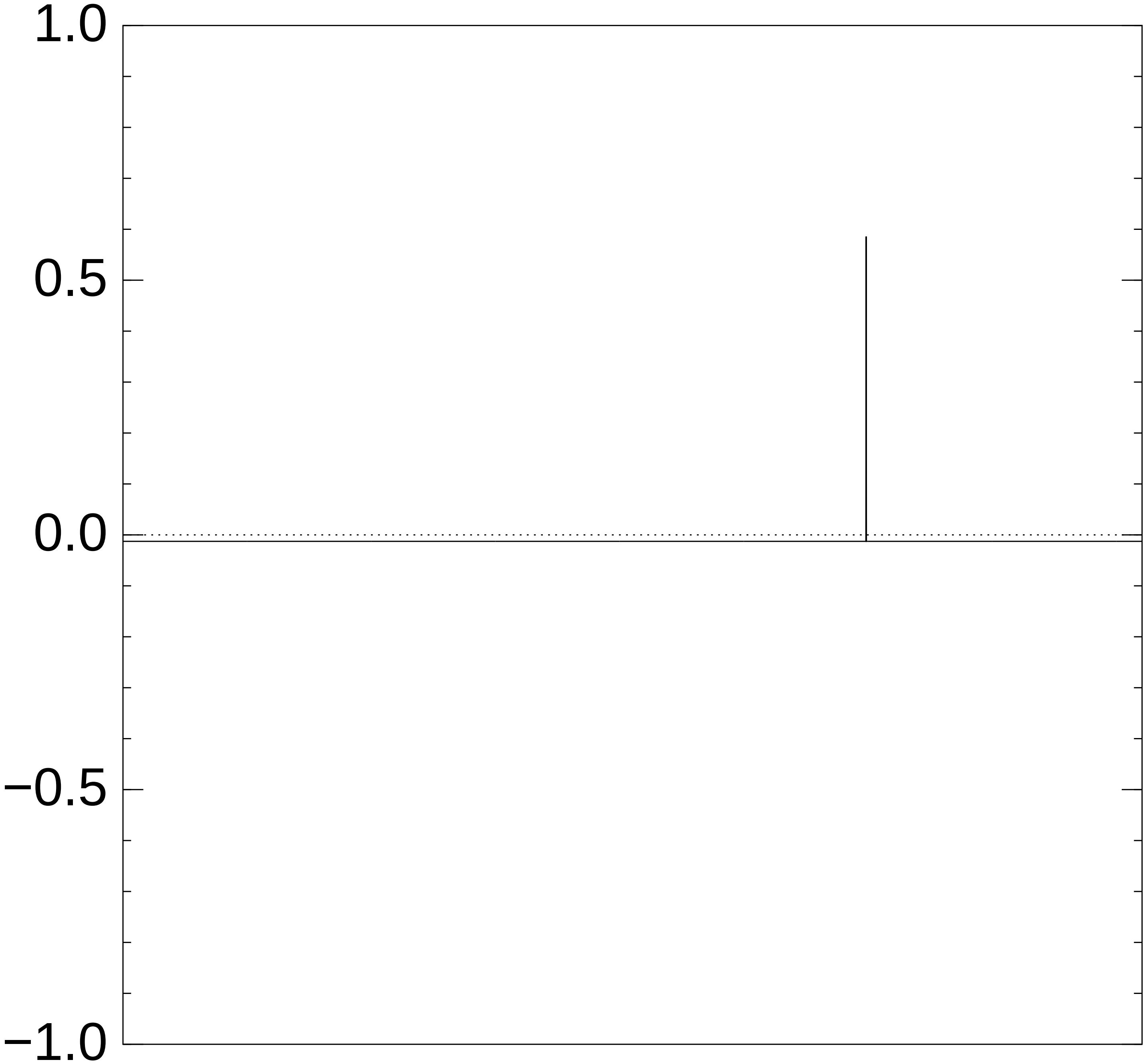
Normalized graph of $q \mapsto a_q$
for an example with $n = 12$
after $60 \times (\text{Step 1} + \text{Step 2})$:



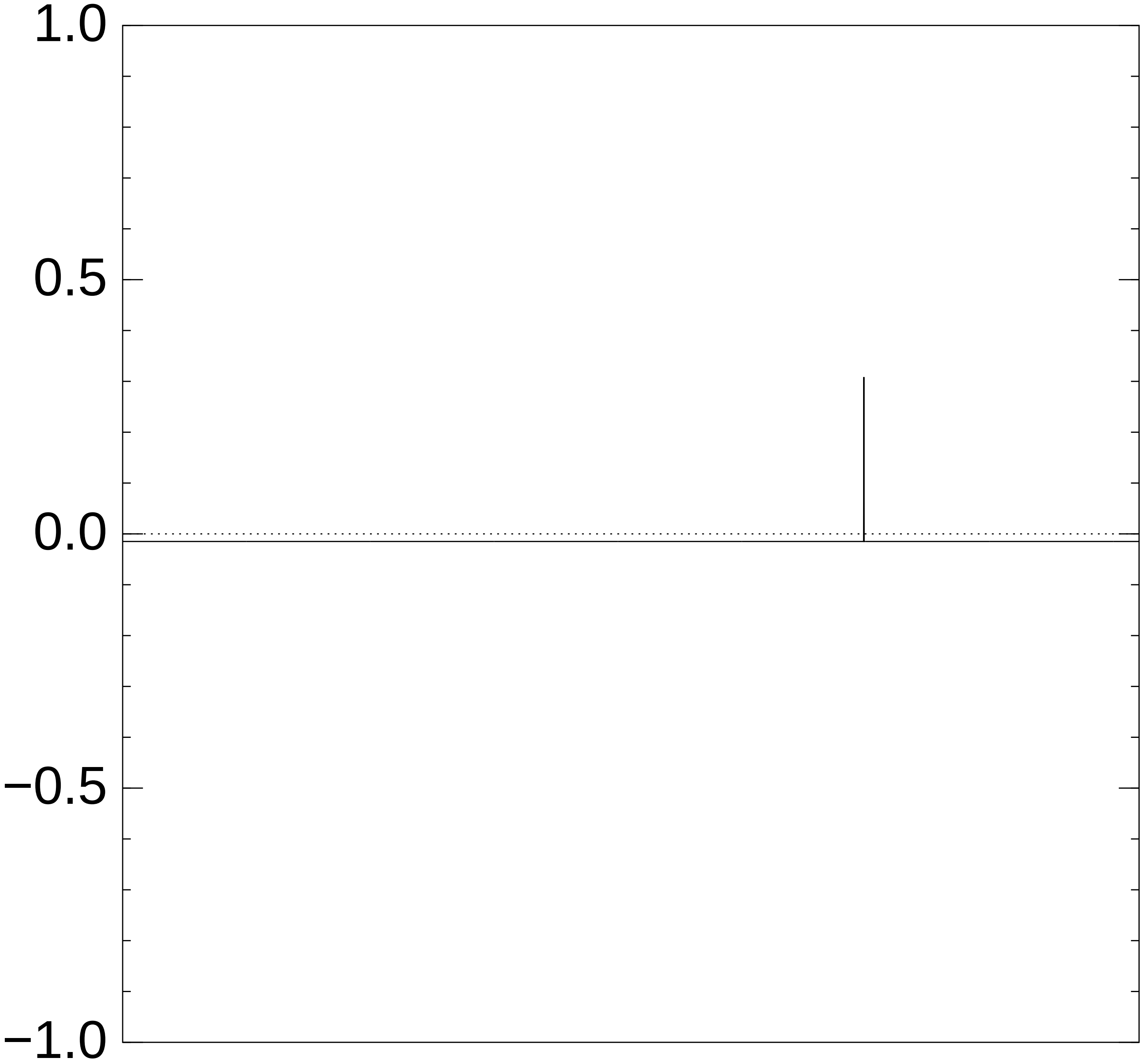
Normalized graph of $q \mapsto a_q$
for an example with $n = 12$
after $70 \times (\text{Step } 1 + \text{Step } 2)$:



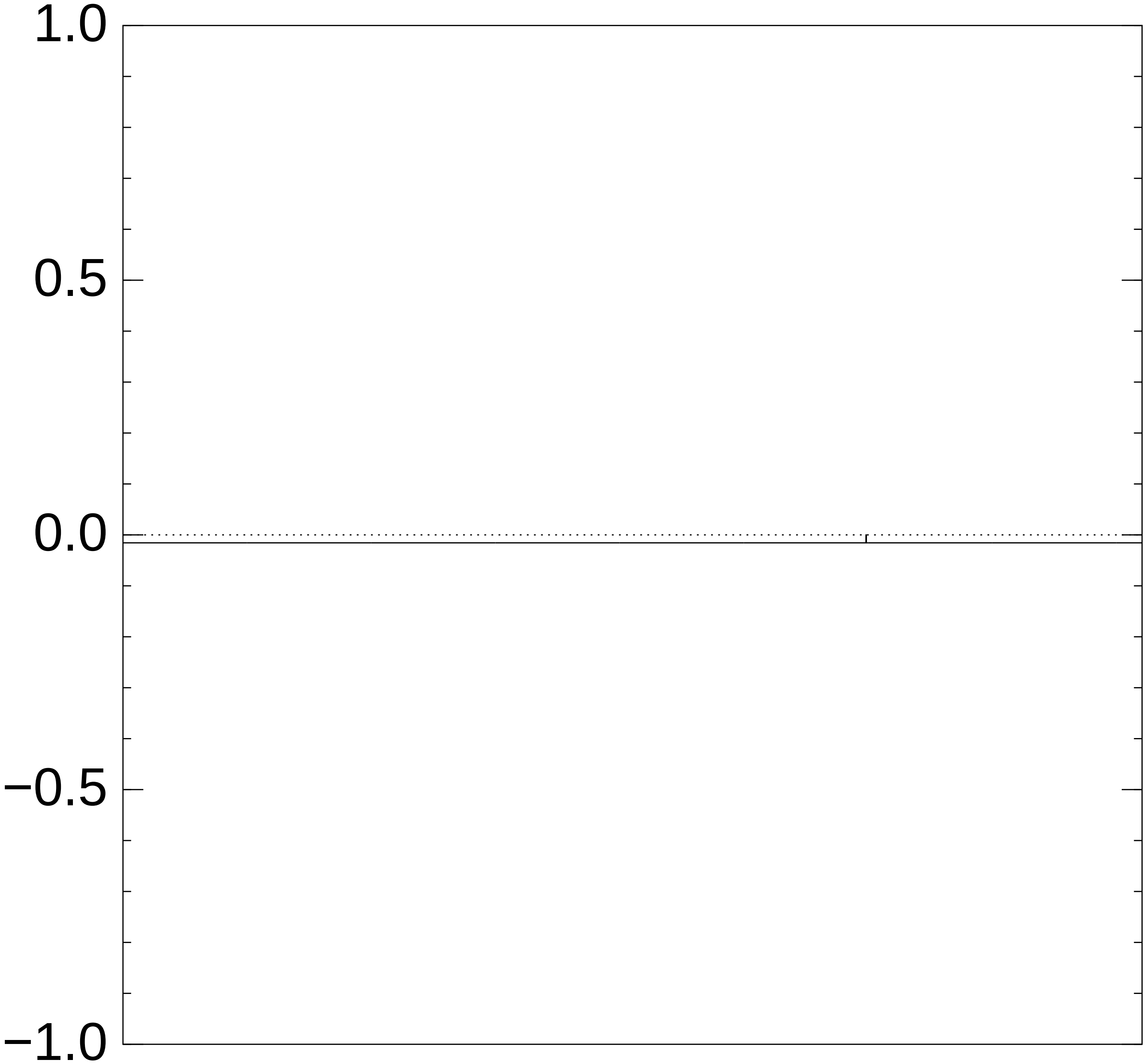
Normalized graph of $q \mapsto a_q$ for an example with $n = 12$ after $80 \times$ (Step $1 +$ Step 2):



Normalized graph of $q \mapsto a_q$
for an example with $n = 12$
after $90 \times$ (Step 1 + Step 2):



Normalized graph of $q \mapsto a_q$ for an example with $n = 12$ after $100 \times$ (Step $1 +$ Step 2):



Very bad stopping point.

$q \mapsto a_q$ is completely described by a vector of two numbers (with fixed multiplicities):

(1) a_q for roots q;

(2) a_q for non-roots q.

$q \mapsto a_q$ is completely described by a vector of two numbers (with fixed multiplicities):
(1) a_q for roots q;
(2) a_q for non-roots q.

Step 1 + Step 2 act linearly on this vector.

$q \mapsto a_q$ is completely described
by a vector of two numbers
(with fixed multiplicities):
(1) a_q for roots q;
(2) a_q for non-roots q.

Step 1 + Step 2
act linearly on this vector.

Easily compute eigenvalues
and powers of this linear map
to understand evolution
of state of Grover's algorithm.
$\Rightarrow$ Probability is $\approx$1
after $\approx(\pi/4)2^{n/2}$ iterations.

Ambainis's algorithm

Unique-collision-finding problem:
Say f has n-bit inputs,
exactly one collision $\{p, q\}$:
i.e., $p \neq q$, $f(p) = f(q)$.
Problem: find this collision.

Ambainis's algorithm

Unique-collision-finding problem:
Say f has n-bit inputs,
exactly one collision $\{p, q\}$:
i.e., $p \neq q$, $f(p) = f(q)$.
Problem: find this collision.

Cost 2^n: Define S as
the set of n-bit strings.
Compute $f(S)$, sort.

Ambainis's algorithm

Unique-collision-finding problem:
Say f has n-bit inputs,
exactly one collision $\{p, q\}$:
i.e., $p \neq q$, $f(p) = f(q)$.
Problem: find this collision.

Cost 2^n: Define S as
the set of n-bit strings.
Compute $f(S)$, sort.

Generalize to cost r,
success probability $\approx(r/2^n)^2$:
Choose a set S of size r.
Compute $f(S)$, sort.

Data structure $D(S)$ capturing the generalized computation: the set S; the multiset $f(S)$; the number of collisions in S.

Data structure $D(S)$ capturing the generalized computation: the set S; the multiset $f(S)$; the number of collisions in S.

Very efficient to move from $D(S)$ to $D(T)$ if T is an **adjacent** set: $\#S = \#T = r$, $\#(S \cap T) = r - 1$.

Data structure $D(S)$ capturing the generalized computation: the set S; the multiset $f(S)$; the number of collisions in S.

Very efficient to move from $D(S)$ to $D(T)$ if T is an **adjacent** set: $\#S = \#T = r$, $\#(S \cap T) = r - 1$.

2003 Ambainis, simplified 2007 Magniez–Nayak–Roland–Santha: Create superposition of states $(D(S), D(T))$ with adjacent S, T. By a quantum walk find S containing a collision.

How the quantum walk works:

Start from uniform superposition.

Repeat $\approx 0.6 \cdot 2^n/r$ times:

Negate $a_{S,T}$

if S contains collision.

Repeat $\approx 0.7 \cdot \sqrt{r}$ times:

For each T:

Diffuse $a_{S,T}$ across all S.

For each S:

Diffuse $a_{S,T}$ across all T.

Now high probability

that T contains collision.

Cost $r + 2^n/\sqrt{r}$. Optimize: $2^{2n/3}$.

Classify (S,T) according to $(\#(S\cap\{p,q\}),\#(T\cap\{p,q\}))$; reduce a to low-dim vector. Analyze evolution of this vector.

e.g. $n=15$, $r=1024$, after 0 negations and 0 diffusions:

$\Pr[\text{class }(0,0)]\approx 0.938;+$
$\Pr[\text{class }(0,1)]\approx 0.000;+$
$\Pr[\text{class }(1,0)]\approx 0.000;+$
$\Pr[\text{class }(1,1)]\approx 0.060;+$
$\Pr[\text{class }(1,2)]\approx 0.000;+$
$\Pr[\text{class }(2,1)]\approx 0.000;+$
$\Pr[\text{class }(2,2)]\approx 0.001;+$

Right column is sign of $a_{S,T}$.

Classify (S, T) according to $(\#(S \cap \{p, q\}), \#(T \cap \{p, q\}))$; reduce a to low-dim vector. Analyze evolution of this vector.

e.g. $n = 15$, $r = 1024$, after 1 negation and 46 diffusions:

$\Pr[\text{class } (0, 0)] \approx 0.935; +$
$\Pr[\text{class } (0, 1)] \approx 0.000; +$
$\Pr[\text{class } (1, 0)] \approx 0.000; -$
$\Pr[\text{class } (1, 1)] \approx 0.057; +$
$\Pr[\text{class } (1, 2)] \approx 0.000; +$
$\Pr[\text{class } (2, 1)] \approx 0.000; -$
$\Pr[\text{class } (2, 2)] \approx 0.008; +$

Right column is sign of $a_{S,T}$.

Classify (S, T) according to $(\#(S \cap \{p, q\}), \#(T \cap \{p, q\}))$; reduce a to low-dim vector. Analyze evolution of this vector.

e.g. $n = 15$, $r = 1024$, after 2 negations and 92 diffusions:

$\Pr[\text{class } (0, 0)] \approx 0.918; +$
$\Pr[\text{class } (0, 1)] \approx 0.001; +$
$\Pr[\text{class } (1, 0)] \approx 0.000; -$
$\Pr[\text{class } (1, 1)] \approx 0.059; +$
$\Pr[\text{class } (1, 2)] \approx 0.001; +$
$\Pr[\text{class } (2, 1)] \approx 0.000; -$
$\Pr[\text{class } (2, 2)] \approx 0.022; +$

Right column is sign of $a_{S,T}$.

Classify (S, T) according to $(\#(S \cap \{p, q\}), \#(T \cap \{p, q\}))$; reduce a to low-dim vector. Analyze evolution of this vector.

e.g. $n = 15$, $r = 1024$, after 3 negations and 138 diffusions:

$\Pr[\text{class } (0, 0)] \approx 0.897; +$
$\Pr[\text{class } (0, 1)] \approx 0.001; +$
$\Pr[\text{class } (1, 0)] \approx 0.000; -$
$\Pr[\text{class } (1, 1)] \approx 0.058; +$
$\Pr[\text{class } (1, 2)] \approx 0.002; +$
$\Pr[\text{class } (2, 1)] \approx 0.000; +$
$\Pr[\text{class } (2, 2)] \approx 0.042; +$

Right column is sign of $a_{S,T}$.

Classify (S, T) according to $(\#(S \cap \{p, q\}), \#(T \cap \{p, q\}))$; reduce a to low-dim vector. Analyze evolution of this vector.

e.g. $n = 15$, $r = 1024$, after 4 negations and 184 diffusions:

$\Pr[\text{class } (0, 0)] \approx 0.873; +$
$\Pr[\text{class } (0, 1)] \approx 0.001; +$
$\Pr[\text{class } (1, 0)] \approx 0.000; -$
$\Pr[\text{class } (1, 1)] \approx 0.054; +$
$\Pr[\text{class } (1, 2)] \approx 0.002; +$
$\Pr[\text{class } (2, 1)] \approx 0.000; +$
$\Pr[\text{class } (2, 2)] \approx 0.070; +$

Right column is sign of $a_{S,T}$.

Classify (S, T) according to $(\#(S \cap \{p, q\}), \#(T \cap \{p, q\}))$; reduce a to low-dim vector. Analyze evolution of this vector.

e.g. $n = 15$, $r = 1024$, after 5 negations and 230 diffusions:

$\Pr[\text{class } (0, 0)] \approx 0.838; +$
$\Pr[\text{class } (0, 1)] \approx 0.001; +$
$\Pr[\text{class } (1, 0)] \approx 0.001; -$
$\Pr[\text{class } (1, 1)] \approx 0.054; +$
$\Pr[\text{class } (1, 2)] \approx 0.003; +$
$\Pr[\text{class } (2, 1)] \approx 0.000; +$
$\Pr[\text{class } (2, 2)] \approx 0.104; +$

Right column is sign of $a_{S,T}$.

Classify (S, T) according to $(\#(S \cap \{p, q\}), \#(T \cap \{p, q\}))$; reduce a to low-dim vector. Analyze evolution of this vector.

e.g. $n = 15$, $r = 1024$, after 6 negations and 276 diffusions:

$\Pr[\text{class } (0, 0)] \approx 0.800; +$
$\Pr[\text{class } (0, 1)] \approx 0.001; +$
$\Pr[\text{class } (1, 0)] \approx 0.001; -$
$\Pr[\text{class } (1, 1)] \approx 0.051; +$
$\Pr[\text{class } (1, 2)] \approx 0.006; +$
$\Pr[\text{class } (2, 1)] \approx 0.000; +$
$\Pr[\text{class } (2, 2)] \approx 0.141; +$

Right column is sign of $a_{S,T}$.

Classify (S, T) according to
$(\#(S \cap \{p, q\}), \#(T \cap \{p, q\}))$;
reduce a to low-dim vector.
Analyze evolution of this vector.

e.g. $n = 15$, $r = 1024$, after
7 negations and 322 diffusions:

$\Pr[\text{class } (0, 0)] \approx 0.758; +$
$\Pr[\text{class } (0, 1)] \approx 0.002; +$
$\Pr[\text{class } (1, 0)] \approx 0.001; -$
$\Pr[\text{class } (1, 1)] \approx 0.047; +$
$\Pr[\text{class } (1, 2)] \approx 0.007; +$
$\Pr[\text{class } (2, 1)] \approx 0.000; +$
$\Pr[\text{class } (2, 2)] \approx 0.184; +$

Right column is sign of $a_{S,T}$.

Classify (S, T) according to
$(\#(S \cap \{p, q\}), \#(T \cap \{p, q\}))$;
reduce a to low-dim vector.
Analyze evolution of this vector.

e.g. $n = 15$, $r = 1024$, after
8 negations and 368 diffusions:

$\Pr[\text{class } (0, 0)] \approx 0.708; +$
$\Pr[\text{class } (0, 1)] \approx 0.003; +$
$\Pr[\text{class } (1, 0)] \approx 0.001; -$
$\Pr[\text{class } (1, 1)] \approx 0.046; +$
$\Pr[\text{class } (1, 2)] \approx 0.007; +$
$\Pr[\text{class } (2, 1)] \approx 0.000; +$
$\Pr[\text{class } (2, 2)] \approx 0.234; +$

Right column is sign of $a_{S,T}$.

Classify (S, T) according to $(\#(S \cap \{p, q\}), \#(T \cap \{p, q\}))$; reduce a to low-dim vector.

Analyze evolution of this vector.

e.g. $n = 15$, $r = 1024$, after 9 negations and 414 diffusions:

$\Pr[\text{class } (0, 0)] \approx 0.658; +$
$\Pr[\text{class } (0, 1)] \approx 0.003; +$
$\Pr[\text{class } (1, 0)] \approx 0.001; -$
$\Pr[\text{class } (1, 1)] \approx 0.042; +$
$\Pr[\text{class } (1, 2)] \approx 0.009; +$
$\Pr[\text{class } (2, 1)] \approx 0.000; +$
$\Pr[\text{class } (2, 2)] \approx 0.287; +$

Right column is sign of $a_{S,T}$.

Classify (S, T) according to
$(\#(S \cap \{p, q\}), \#(T \cap \{p, q\}))$;
reduce a to low-dim vector.
Analyze evolution of this vector.

e.g. $n = 15$, $r = 1024$, after
10 negations and 460 diffusions:

$\Pr[\text{class } (0, 0)] \approx 0.606; +$
$\Pr[\text{class } (0, 1)] \approx 0.003; +$
$\Pr[\text{class } (1, 0)] \approx 0.002; -$
$\Pr[\text{class } (1, 1)] \approx 0.037; +$
$\Pr[\text{class } (1, 2)] \approx 0.013; +$
$\Pr[\text{class } (2, 1)] \approx 0.000; +$
$\Pr[\text{class } (2, 2)] \approx 0.338; +$

Right column is sign of $a_{S,T}$.

Classify (S, T) according to $(\#(S \cap \{p, q\}), \#(T \cap \{p, q\}))$; reduce a to low-dim vector. Analyze evolution of this vector.

e.g. $n = 15$, $r = 1024$, after 11 negations and 506 diffusions:

$\Pr[\text{class } (0, 0)] \approx 0.547; +$
$\Pr[\text{class } (0, 1)] \approx 0.004; +$
$\Pr[\text{class } (1, 0)] \approx 0.003; -$
$\Pr[\text{class } (1, 1)] \approx 0.036; +$
$\Pr[\text{class } (1, 2)] \approx 0.015; +$
$\Pr[\text{class } (2, 1)] \approx 0.001; +$
$\Pr[\text{class } (2, 2)] \approx 0.394; +$

Right column is sign of $a_{S,T}$.

Classify (S, T) according to $(\#(S \cap \{p, q\}), \#(T \cap \{p, q\}))$; reduce a to low-dim vector. Analyze evolution of this vector.

e.g. $n = 15$, $r = 1024$, after 12 negations and 552 diffusions:

$\Pr[\text{class } (0, 0)] \approx 0.491; +$
$\Pr[\text{class } (0, 1)] \approx 0.004; +$
$\Pr[\text{class } (1, 0)] \approx 0.003; -$
$\Pr[\text{class } (1, 1)] \approx 0.032; +$
$\Pr[\text{class } (1, 2)] \approx 0.014; +$
$\Pr[\text{class } (2, 1)] \approx 0.001; +$
$\Pr[\text{class } (2, 2)] \approx 0.455; +$

Right column is sign of $a_{S,T}$.

Classify (S, T) according to $(\#(S \cap \{p, q\}), \#(T \cap \{p, q\}))$; reduce a to low-dim vector. Analyze evolution of this vector.

e.g. $n = 15$, $r = 1024$, after 13 negations and 598 diffusions:

$\Pr[\text{class } (0, 0)] \approx 0.436; +$
$\Pr[\text{class } (0, 1)] \approx 0.005; +$
$\Pr[\text{class } (1, 0)] \approx 0.003; -$
$\Pr[\text{class } (1, 1)] \approx 0.026; +$
$\Pr[\text{class } (1, 2)] \approx 0.017; +$
$\Pr[\text{class } (2, 1)] \approx 0.000; +$
$\Pr[\text{class } (2, 2)] \approx 0.513; +$

Right column is sign of $a_{S,T}$.

Classify (S, T) according to $(\#(S \cap \{p, q\}), \#(T \cap \{p, q\}))$; reduce a to low-dim vector. Analyze evolution of this vector.

e.g. $n = 15$, $r = 1024$, after 14 negations and 644 diffusions:

$\Pr[\text{class } (0, 0)] \approx 0.377; +$
$\Pr[\text{class } (0, 1)] \approx 0.006; +$
$\Pr[\text{class } (1, 0)] \approx 0.004; -$
$\Pr[\text{class } (1, 1)] \approx 0.025; +$
$\Pr[\text{class } (1, 2)] \approx 0.022; +$
$\Pr[\text{class } (2, 1)] \approx 0.001; +$
$\Pr[\text{class } (2, 2)] \approx 0.566; +$

Right column is sign of $a_{S,T}$.

Classify (S, T) according to $(\#(S \cap \{p, q\}), \#(T \cap \{p, q\}))$; reduce a to low-dim vector. Analyze evolution of this vector.

e.g. $n = 15$, $r = 1024$, after 15 negations and 690 diffusions:

$\Pr[\text{class } (0, 0)] \approx 0.322; +$
$\Pr[\text{class } (0, 1)] \approx 0.005; +$
$\Pr[\text{class } (1, 0)] \approx 0.004; -$
$\Pr[\text{class } (1, 1)] \approx 0.021; +$
$\Pr[\text{class } (1, 2)] \approx 0.023; +$
$\Pr[\text{class } (2, 1)] \approx 0.001; +$
$\Pr[\text{class } (2, 2)] \approx 0.623; +$

Right column is sign of $a_{S,T}$.

Classify (S, T) according to $(\#(S \cap \{p, q\}), \#(T \cap \{p, q\}))$; reduce a to low-dim vector. Analyze evolution of this vector.

e.g. $n = 15$, $r = 1024$, after 16 negations and 736 diffusions:

$\Pr[\text{class } (0, 0)] \approx 0.270; +$
$\Pr[\text{class } (0, 1)] \approx 0.006; +$
$\Pr[\text{class } (1, 0)] \approx 0.005; -$
$\Pr[\text{class } (1, 1)] \approx 0.017; +$
$\Pr[\text{class } (1, 2)] \approx 0.022; +$
$\Pr[\text{class } (2, 1)] \approx 0.001; +$
$\Pr[\text{class } (2, 2)] \approx 0.680; +$

Right column is sign of $a_{S,T}$.

Classify (S, T) according to $(\#(S \cap \{p, q\}), \#(T \cap \{p, q\}))$; reduce a to low-dim vector. Analyze evolution of this vector.

e.g. $n = 15$, $r = 1024$, after 17 negations and 782 diffusions:

$\Pr[\text{class } (0, 0)] \approx 0.218; +$
$\Pr[\text{class } (0, 1)] \approx 0.007; +$
$\Pr[\text{class } (1, 0)] \approx 0.005; -$
$\Pr[\text{class } (1, 1)] \approx 0.015; +$
$\Pr[\text{class } (1, 2)] \approx 0.024; +$
$\Pr[\text{class } (2, 1)] \approx 0.001; +$
$\Pr[\text{class } (2, 2)] \approx 0.730; +$

Right column is sign of $a_{S,T}$.

Classify (S, T) according to
$(\#(S \cap \{p, q\}), \#(T \cap \{p, q\}))$;
reduce a to low-dim vector.
Analyze evolution of this vector.

e.g. $n = 15$, $r = 1024$, after
18 negations and 828 diffusions:

$\Pr[\text{class } (0, 0)] \approx 0.172; +$
$\Pr[\text{class } (0, 1)] \approx 0.006; +$
$\Pr[\text{class } (1, 0)] \approx 0.005; -$
$\Pr[\text{class } (1, 1)] \approx 0.011; +$
$\Pr[\text{class } (1, 2)] \approx 0.029; +$
$\Pr[\text{class } (2, 1)] \approx 0.001; +$
$\Pr[\text{class } (2, 2)] \approx 0.775; +$

Right column is sign of $a_{S,T}$.

Classify (S, T) according to $(\#(S \cap \{p, q\}), \#(T \cap \{p, q\}))$; reduce a to low-dim vector. Analyze evolution of this vector.

e.g. $n = 15$, $r = 1024$, after 19 negations and 874 diffusions:

$\Pr[\text{class } (0, 0)] \approx 0.131; +$
$\Pr[\text{class } (0, 1)] \approx 0.007; +$
$\Pr[\text{class } (1, 0)] \approx 0.006; -$
$\Pr[\text{class } (1, 1)] \approx 0.008; +$
$\Pr[\text{class } (1, 2)] \approx 0.030; +$
$\Pr[\text{class } (2, 1)] \approx 0.002; +$
$\Pr[\text{class } (2, 2)] \approx 0.816; +$

Right column is sign of $a_{S,T}$.

Classify (S, T) according to
$(\#(S \cap \{p, q\}), \#(T \cap \{p, q\}))$;
reduce a to low-dim vector.
Analyze evolution of this vector.

e.g. $n = 15$, $r = 1024$, after
20 negations and 920 diffusions:

$\Pr[\text{class } (0, 0)] \approx 0.093; +$
$\Pr[\text{class } (0, 1)] \approx 0.007; +$
$\Pr[\text{class } (1, 0)] \approx 0.007; -$
$\Pr[\text{class } (1, 1)] \approx 0.007; +$
$\Pr[\text{class } (1, 2)] \approx 0.027; +$
$\Pr[\text{class } (2, 1)] \approx 0.002; +$
$\Pr[\text{class } (2, 2)] \approx 0.857; +$

Right column is sign of $a_{S,T}$.

Classify (S, T) according to $(\#(S \cap \{p, q\}), \#(T \cap \{p, q\}))$; reduce a to low-dim vector. Analyze evolution of this vector.

e.g. $n = 15$, $r = 1024$, after 21 negations and 966 diffusions:

$\Pr[\text{class } (0, 0)] \approx 0.062; +$
$\Pr[\text{class } (0, 1)] \approx 0.007; +$
$\Pr[\text{class } (1, 0)] \approx 0.006; -$
$\Pr[\text{class } (1, 1)] \approx 0.004; +$
$\Pr[\text{class } (1, 2)] \approx 0.030; +$
$\Pr[\text{class } (2, 1)] \approx 0.001; +$
$\Pr[\text{class } (2, 2)] \approx 0.890; +$

Right column is sign of $a_{S,T}$.

Classify (S, T) according to $(\#(S \cap \{p, q\}), \#(T \cap \{p, q\}))$; reduce a to low-dim vector. Analyze evolution of this vector.

e.g. $n = 15$, $r = 1024$, after 22 negations and 1012 diffusions:

$\Pr[\text{class } (0, 0)] \approx 0.037; +$
$\Pr[\text{class } (0, 1)] \approx 0.008; +$
$\Pr[\text{class } (1, 0)] \approx 0.007; -$
$\Pr[\text{class } (1, 1)] \approx 0.002; +$
$\Pr[\text{class } (1, 2)] \approx 0.034; +$
$\Pr[\text{class } (2, 1)] \approx 0.001; +$
$\Pr[\text{class } (2, 2)] \approx 0.910; +$

Right column is sign of $a_{S,T}$.

Classify (S, T) according to $(\#(S \cap \{p, q\}), \#(T \cap \{p, q\}))$; reduce a to low-dim vector. Analyze evolution of this vector.

e.g. $n = 15$, $r = 1024$, after 23 negations and 1058 diffusions:

$\Pr[\text{class } (0, 0)] \approx 0.017; +$
$\Pr[\text{class } (0, 1)] \approx 0.008; +$
$\Pr[\text{class } (1, 0)] \approx 0.007; -$
$\Pr[\text{class } (1, 1)] \approx 0.002; +$
$\Pr[\text{class } (1, 2)] \approx 0.034; +$
$\Pr[\text{class } (2, 1)] \approx 0.002; +$
$\Pr[\text{class } (2, 2)] \approx 0.930; +$

Right column is sign of $a_{S,T}$.

Classify (S, T) according to $(\#(S \cap \{p, q\}), \#(T \cap \{p, q\}))$; reduce a to low-dim vector. Analyze evolution of this vector.

e.g. $n = 15$, $r = 1024$, after 24 negations and 1104 diffusions:

$\Pr[\text{class } (0, 0)] \approx 0.005; +$
$\Pr[\text{class } (0, 1)] \approx 0.007; +$
$\Pr[\text{class } (1, 0)] \approx 0.007; -$
$\Pr[\text{class } (1, 1)] \approx 0.000; +$
$\Pr[\text{class } (1, 2)] \approx 0.030; +$
$\Pr[\text{class } (2, 1)] \approx 0.002; +$
$\Pr[\text{class } (2, 2)] \approx 0.948; +$

Right column is sign of $a_{S,T}$.

Classify (S, T) according to $(\#(S \cap \{p, q\}), \#(T \cap \{p, q\}))$; reduce a to low-dim vector. Analyze evolution of this vector.

e.g. $n = 15$, $r = 1024$, after 25 negations and 1150 diffusions:

$\Pr[\text{class } (0, 0)] \approx 0.000; +$
$\Pr[\text{class } (0, 1)] \approx 0.008; +$
$\Pr[\text{class } (1, 0)] \approx 0.008; -$
$\Pr[\text{class } (1, 1)] \approx 0.000; +$
$\Pr[\text{class } (1, 2)] \approx 0.031; +$
$\Pr[\text{class } (2, 1)] \approx 0.001; +$
$\Pr[\text{class } (2, 2)] \approx 0.952; +$

Right column is sign of $a_{S,T}$.

Classify (S, T) according to $(\#(S \cap \{p, q\}), \#(T \cap \{p, q\}))$; reduce a to low-dim vector. Analyze evolution of this vector.

e.g. $n = 15$, $r = 1024$, after 26 negations and 1196 diffusions:

$\Pr[\text{class } (0, 0)] \approx 0.002; -$
$\Pr[\text{class } (0, 1)] \approx 0.008; +$
$\Pr[\text{class } (1, 0)] \approx 0.008; -$
$\Pr[\text{class } (1, 1)] \approx 0.000; -$
$\Pr[\text{class } (1, 2)] \approx 0.035; +$
$\Pr[\text{class } (2, 1)] \approx 0.002; +$
$\Pr[\text{class } (2, 2)] \approx 0.945; +$

Right column is sign of $a_{S,T}$.

Classify (S, T) according to
$(\#(S \cap \{p, q\}), \#(T \cap \{p, q\}))$;
reduce a to low-dim vector.
Analyze evolution of this vector.

e.g. $n = 15$, $r = 1024$, after
27 negations and 1242 diffusions:

$\Pr[\text{class } (0, 0)] \approx 0.011; -$
$\Pr[\text{class } (0, 1)] \approx 0.007; +$
$\Pr[\text{class } (1, 0)] \approx 0.007; -$
$\Pr[\text{class } (1, 1)] \approx 0.001; -$
$\Pr[\text{class } (1, 2)] \approx 0.034; +$
$\Pr[\text{class } (2, 1)] \approx 0.003; +$
$\Pr[\text{class } (2, 2)] \approx 0.938; +$

Right column is sign of $a_{S,T}$.

Data structures

Moving from $D(S)$ to $D(T)$: dominated by $O(1)$ evaluations of f if f is extremely slow.

But usually f is not so slow.

Data structures

Moving from $D(S)$ to $D(T)$: dominated by $O(1)$ evaluations of f if f is extremely slow.

But usually f is not so slow. Store set S and multiset $f(S)$ in, e.g., hash tables?

Data structures

Moving from $D(S)$ to $D(T)$: dominated by $O(1)$ evaluations of f if f is extremely slow.

But usually f is not so slow. Store set S and multiset $f(S)$ in, e.g., hash tables?

Minor problem: time to hash S is huge for some sets S.

Data structures

Moving from $D(S)$ to $D(T)$: dominated by $O(1)$ evaluations of f if f is extremely slow.

But usually f is not so slow. Store set S and multiset $f(S)$ in, e.g., hash tables?

Minor problem: time to hash S is huge for some sets S.

Fix: randomize hash function (1979 Carter–Wegman), and specify big enough time for whole algorithm to be reliable.

Major problem: hash table depends on history, not just on S. Algorithm fails horribly.

Need history-independent $D(S)$.

Major problem: hash table depends on history, not just on S. Algorithm fails horribly.

Need history-independent $D(S)$.

2003 Ambainis: "combination of a hash table and a skip list". Several pages of analysis.

Major problem: hash table depends on history, not just on S. Algorithm fails horribly.

Need history-independent $D(S)$.

2003 Ambainis: "combination of a hash table and a skip list". Several pages of analysis.

2013 Bernstein–Jeffery–Lange–Meurer: radix tree.

Simplest radix tree: Left subtree stores $\{x : (0, x) \in S\}$ if nonempty. Right subtree stores $\{x : (1, x) \in S\}$ if nonempty.

Caveats

The $2^{2n/3}$ analysis assumes cheap random access to memory. Justified by simplicity, not realism.

Caveats

The $2^{2n/3}$ analysis assumes cheap random access to memory. Justified by simplicity, not realism.

Can we move data using energy sublinear in distance moved?

Caveats

The $2^{2n/3}$ analysis assumes cheap random access to memory. Justified by simplicity, not realism.

Can we move data using energy sublinear in distance moved?
2015 Intel presentation says that moving 8 bytes on wire at 22nm costs 11.20 pJ per 5mm.

Caveats

The $2^{2n/3}$ analysis assumes cheap random access to memory. Justified by simplicity, not realism.

Can we move data using energy sublinear in distance moved? 2015 Intel presentation says that moving 8 bytes on wire at 22nm costs 11.20 pJ per 5mm. Lasers spread. Fibers lose. etc.

Caveats

The $2^{2n/3}$ analysis assumes cheap random access to memory. Justified by simplicity, not realism.

Can we move data using energy sublinear in distance moved? 2015 Intel presentation says that moving 8 bytes on wire at 22nm costs 11.20 pJ per 5mm. Lasers spread. Fibers lose. etc.

I recommend algorithm analysis on 2-dim mesh of tiny processors: e.g. 0.472 for MQ (vs. 0.462) from 2017 Bernstein–Yang.

Many claimed quantum speedups don't seem to exist in this model.

e.g. 2009 Bernstein analysis: fastest algorithm known for random-collision search is 1994 van Oorschot–Wiener.

Many claimed quantum speedups don't seem to exist in this model. e.g. 2009 Bernstein analysis: fastest algorithm known for random-collision search is 1994 van Oorschot–Wiener.

Further obstacles to Grover:

- Parallelization reduces speedup. $D\times$ speedup needs depth D.

Many claimed quantum speedups don't seem to exist in this model. e.g. 2009 Bernstein analysis: fastest algorithm known for random-collision search is 1994 van Oorschot–Wiener.

Further obstacles to Grover:

- Parallelization reduces speedup. $D\times$ speedup needs depth D.
- Reversibility is expensive.

Many claimed quantum speedups don't seem to exist in this model. e.g. 2009 Bernstein analysis: fastest algorithm known for random-collision search is 1994 van Oorschot–Wiener.

Further obstacles to Grover:

- Parallelization reduces speedup. $D\times$ speedup needs depth D.
- Reversibility is expensive.
- Quantum ops are expensive.

Many claimed quantum speedups don't seem to exist in this model.

e.g. 2009 Bernstein analysis: fastest algorithm known for random-collision search is 1994 van Oorschot–Wiener.

Further obstacles to Grover:

- Parallelization reduces speedup. $D\times$ speedup needs depth D.
- Reversibility is expensive.
- Quantum ops are expensive.

Grover risk to cryptography is much smaller than Shor risk.

Background slides . . .

What do quantum computers do?

"Quantum algorithm" means an algorithm that a quantum computer can run.

i.e. a sequence of instructions, where each instruction is in a quantum computer's supported instruction set.

How do we know which instructions a quantum computer will support?

Quantum computer type 1 (QC1):
contains many "qubits";
can efficiently perform
"NOT gate", "Hadamard gate",
"controlled NOT gate", "T gate".

Quantum computer type 1 (QC1):
contains many “qubits”;
can efficiently perform
“NOT gate”, “Hadamard gate”,
“controlled NOT gate”, “*T* gate”.

Making these instructions work is the main goal of quantum-computer engineering.

Quantum computer type 1 (QC1): contains many "qubits"; can efficiently perform "NOT gate", "Hadamard gate", "controlled NOT gate", "*T* gate".

Making these instructions work is the main goal of quantum-computer engineering.

Combine these instructions to compute "Toffoli gate"; . . . "Simon's algorithm"; . . . "Shor's algorithm"; etc.

Quantum computer type 1 (QC1):
contains many “qubits”;
can efficiently perform
“NOT gate”, “Hadamard gate”,
“controlled NOT gate”, “*T* gate”.

Making these instructions work is the main goal of quantum-computer engineering.

Combine these instructions
to compute “Toffoli gate”;
. . . “Simon’s algorithm”;
. . . “Shor’s algorithm”; etc.

General belief: Traditional CPU
isn’t QC1; e.g. can’t factor quickly.

Quantum computer type 2 (QC2): stores a simulated universe; efficiently simulates the laws of quantum physics with as much accuracy as desired.

This is the original concept of quantum computers introduced by 1982 Feynman "Simulating physics with computers".

Quantum computer type 2 (QC2): stores a simulated universe; efficiently simulates the laws of quantum physics with as much accuracy as desired.

This is the original concept of quantum computers introduced by 1982 Feynman "Simulating physics with computers".

General belief: any QC1 is a QC2. Partial proof: see, e.g., 2011 Jordan–Lee–Preskill "Quantum algorithms for quantum field theories".

Quantum computer type 3 (QC3):
efficiently computes anything
that any possible physical
computer can compute efficiently.

Quantum computer type 3 (QC3): efficiently computes anything that any possible physical computer can compute efficiently.

General belief: any QC2 is a QC3.

Argument for belief: any physical computer must follow the laws of quantum physics, so a QC2 can efficiently simulate any physical computer.

Quantum computer type 3 (QC3): efficiently computes anything that any possible physical computer can compute efficiently.

General belief: any QC2 is a QC3.
Argument for belief:
any physical computer must follow the laws of quantum physics, so a QC2 can efficiently simulate any physical computer.

General belief: any QC3 is a QC1.
Argument for belief:
look, we're building a QC1.

A note on D-Wave

Apparent scientific consensus:
Current "quantum computers" from D-Wave are useless— can be more cost-effectively simulated by traditional CPUs.

A note on D-Wave

Apparent scientific consensus:
Current “quantum computers”
from D-Wave are useless—
can be more cost-effectively
simulated by traditional CPUs.

But D-Wave is

- collecting venture capital;

A note on D-Wave

Apparent scientific consensus: Current "quantum computers" from D-Wave are useless— can be more cost-effectively simulated by traditional CPUs.

But D-Wave is

- collecting venture capital;
- selling some machines;

A note on D-Wave

Apparent scientific consensus: Current "quantum computers" from D-Wave are useless— can be more cost-effectively simulated by traditional CPUs.

But D-Wave is

- collecting venture capital;
- selling some machines;
- collecting possibly useful engineering expertise;

A note on D-Wave

Apparent scientific consensus:
Current "quantum computers"
from D-Wave are useless—
can be more cost-effectively
simulated by traditional CPUs.

But D-Wave is

- collecting venture capital;
- selling some machines;
- collecting possibly useful engineering expertise;
- not being punished for deceiving people.

A note on D-Wave

Apparent scientific consensus: Current "quantum computers" from D-Wave are useless— can be more cost-effectively simulated by traditional CPUs.

But D-Wave is

- collecting venture capital;
- selling some machines;
- collecting possibly useful engineering expertise;
- not being punished for deceiving people.

Is D-Wave a bad investment?

The state of a computer

Data ("state") stored in 3 bits:
a list of 3 elements of $\{0, 1\}$.
e.g.: $(0, 0, 0)$.

The state of a computer

Data (“state”) stored in 3 bits:
a list of 3 elements of $\{0, 1\}$.
e.g.: $(0, 0, 0)$.
e.g.: $(1, 1, 1)$.

The state of a computer

Data ("state") stored in 3 bits:
a list of 3 elements of $\{0, 1\}$.

e.g.: $(0, 0, 0)$.

e.g.: $(1, 1, 1)$.

e.g.: $(0, 1, 1)$.

The state of a computer

Data ("state") stored in 3 bits:
a list of 3 elements of $\{0,1\}$.
e.g.: $(0,0,0)$.
e.g.: $(1,1,1)$.
e.g.: $(0,1,1)$.

Data stored in 64 bits:
a list of 64 elements of $\{0,1\}$.

The state of a computer

Data ("state") stored in 3 bits:
a list of 3 elements of $\{0, 1\}$.
e.g.: $(0, 0, 0)$.
e.g.: $(1, 1, 1)$.
e.g.: $(0, 1, 1)$.

Data stored in 64 bits:
a list of 64 elements of $\{0, 1\}$.
e.g.: $(1, 1, 1, 1, 1, 0, 0, 0, 1,$
$0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,$
$0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,$
$1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,$
$0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,$
$1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1)$.

The state of a quantum computer

Data stored in 3 qubits:
a list of 8 numbers, not all zero.
e.g.: $(3, 1, 4, 1, 5, 9, 2, 6)$.

The state of a quantum computer

Data stored in 3 qubits:
a list of 8 numbers, not all zero.
e.g.: $(3, 1, 4, 1, 5, 9, 2, 6)$.
e.g.: $(-2, 7, -1, 8, 1, -8, -2, 8)$.

The state of a quantum computer

Data stored in 3 qubits:

a list of 8 numbers, not all zero.

e.g.: $(3, 1, 4, 1, 5, 9, 2, 6)$.

e.g.: $(-2, 7, -1, 8, 1, -8, -2, 8)$.

e.g.: $(0, 0, 0, 0, 0, 1, 0, 0)$.

The state of a quantum computer

Data stored in 3 qubits:
a list of 8 numbers, not all zero.
e.g.: $(3, 1, 4, 1, 5, 9, 2, 6)$.
e.g.: $(-2, 7, -1, 8, 1, -8, -2, 8)$.
e.g.: $(0, 0, 0, 0, 0, 1, 0, 0)$.

Data stored in 4 qubits: a list of 16 numbers, not all zero. e.g.:
$(3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5, 8, 9, 7, 9, 3)$.

The state of a quantum computer

Data stored in 3 qubits:
a list of 8 numbers, not all zero.
e.g.: $(3, 1, 4, 1, 5, 9, 2, 6)$.
e.g.: $(-2, 7, -1, 8, 1, -8, -2, 8)$.
e.g.: $(0, 0, 0, 0, 0, 1, 0, 0)$.

Data stored in 4 qubits: a list of 16 numbers, not all zero. e.g.:
$(3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5, 8, 9, 7, 9, 3)$.

Data stored in 64 qubits:
a list of 2^{64} numbers, not all zero.

The state of a quantum computer

Data stored in 3 qubits:
a list of 8 numbers, not all zero.
e.g.: $(3, 1, 4, 1, 5, 9, 2, 6)$.
e.g.: $(-2, 7, -1, 8, 1, -8, -2, 8)$.
e.g.: $(0, 0, 0, 0, 0, 1, 0, 0)$.

Data stored in 4 qubits: a list of 16 numbers, not all zero. e.g.:
$(3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5, 8, 9, 7, 9, 3)$.

Data stored in 64 qubits:
a list of 2^{64} numbers, not all zero.

Data stored in 1000 qubits: a list of 2^{1000} numbers, not all zero.

Measuring a quantum computer

Can simply look at a bit.

Cannot simply look at the list of numbers stored in n qubits.

Measuring a quantum computer

Can simply look at a bit.
Cannot simply look at the list
of numbers stored in n qubits.

Measuring n qubits

- produces n bits and
- destroys the state.

Measuring a quantum computer

Can simply look at a bit.
Cannot simply look at the list of numbers stored in n qubits.

Measuring n qubits

- produces n bits and
- destroys the state.

If n qubits have state $(a_0, a_1, \ldots, a_{2^n-1})$ then measurement produces q with probability $|a_q|^2 / \sum_r |a_r|^2$.

Measuring a quantum computer

Can simply look at a bit.
Cannot simply look at the list of numbers stored in n qubits.

Measuring n qubits

- produces n bits and
- destroys the state.

If n qubits have state $(a_0, a_1, \ldots, a_{2^n-1})$ then measurement produces q with probability $|a_q|^2 / \sum_r |a_r|^2$.

State is then all zeros except 1 at position q.

e.g.: Say 3 qubits have state
$(1, 1, 1, 1, 1, 1, 1, 1)$.

e.g.: Say 3 qubits have state $(1, 1, 1, 1, 1, 1, 1, 1)$.

Measurement produces
$000 = 0$ with probability $1/8$;
$001 = 1$ with probability $1/8$;
$010 = 2$ with probability $1/8$;
$011 = 3$ with probability $1/8$;
$100 = 4$ with probability $1/8$;
$101 = 5$ with probability $1/8$;
$110 = 6$ with probability $1/8$;
$111 = 7$ with probability $1/8$.

e.g.: Say 3 qubits have state $(1, 1, 1, 1, 1, 1, 1, 1)$.

Measurement produces
$000 = 0$ with probability $1/8$;
$001 = 1$ with probability $1/8$;
$010 = 2$ with probability $1/8$;
$011 = 3$ with probability $1/8$;
$100 = 4$ with probability $1/8$;
$101 = 5$ with probability $1/8$;
$110 = 6$ with probability $1/8$;
$111 = 7$ with probability $1/8$.

"Quantum RNG."

e.g.: Say 3 qubits have state $(1,1,1,1,1,1,1,1)$.

Measurement produces
$000 = 0$ with probability $1/8$;
$001 = 1$ with probability $1/8$;
$010 = 2$ with probability $1/8$;
$011 = 3$ with probability $1/8$;
$100 = 4$ with probability $1/8$;
$101 = 5$ with probability $1/8$;
$110 = 6$ with probability $1/8$;
$111 = 7$ with probability $1/8$.

“Quantum RNG.”

Warning: Quantum RNGs sold today are measurably biased.

e.g.: Say 3 qubits have state $(3, 1, 4, 1, 5, 9, 2, 6)$.

e.g.: Say 3 qubits have state $(3, 1, 4, 1, 5, 9, 2, 6)$.

Measurement produces
$000 = 0$ with probability $9/173$;
$001 = 1$ with probability $1/173$;
$010 = 2$ with probability $16/173$;
$011 = 3$ with probability $1/173$;
$100 = 4$ with probability $25/173$;
$101 = 5$ with probability $81/173$;
$110 = 6$ with probability $4/173$;
$111 = 7$ with probability $36/173$.

e.g.: Say 3 qubits have state $(3, 1, 4, 1, 5, 9, 2, 6)$.

Measurement produces
$000 = 0$ with probability $9/173$;
$001 = 1$ with probability $1/173$;
$010 = 2$ with probability $16/173$;
$011 = 3$ with probability $1/173$;
$100 = 4$ with probability $25/173$;
$101 = 5$ with probability $81/173$;
$110 = 6$ with probability $4/173$;
$111 = 7$ with probability $36/173$.

5 is most likely outcome.

e.g.: Say 3 qubits have state
$(0, 0, 0, 0, 0, 1, 0, 0)$.

e.g.: Say 3 qubits have state $(0, 0, 0, 0, 0, 1, 0, 0)$.

Measurement produces
$000 = 0$ with probability 0;
$001 = 1$ with probability 0;
$010 = 2$ with probability 0;
$011 = 3$ with probability 0;
$100 = 4$ with probability 0;
$101 = 5$ with probability 1;
$110 = 6$ with probability 0;
$111 = 7$ with probability 0.

e.g.: Say 3 qubits have state $(0, 0, 0, 0, 0, 1, 0, 0)$.

Measurement produces
$000 = 0$ with probability 0;
$001 = 1$ with probability 0;
$010 = 2$ with probability 0;
$011 = 3$ with probability 0;
$100 = 4$ with probability 0;
$101 = 5$ with probability 1;
$110 = 6$ with probability 0;
$111 = 7$ with probability 0.

5 is guaranteed outcome.

NOT gates

NOT_0 gate on 3 qubits:
$(3, 1, 4, 1, 5, 9, 2, 6) \mapsto$
$(1, 3, 1, 4, 9, 5, 6, 2)$.

NOT gates

NOT_0 gate on 3 qubits:
$(3, 1, 4, 1, 5, 9, 2, 6) \mapsto$
$(1, 3, 1, 4, 9, 5, 6, 2)$.

NOT_0 gate on 4 qubits:
$(3,1,4,1,5,9,2,6,5,3,5,8,9,7,9,3) \mapsto$
$(1,3,1,4,9,5,6,2,3,5,8,5,7,9,3,9)$.

NOT gates

NOT_0 gate on 3 qubits:
$(3, 1, 4, 1, 5, 9, 2, 6) \mapsto$
$(1, 3, 1, 4, 9, 5, 6, 2)$.

NOT_0 gate on 4 qubits:
$(3,1,4,1,5,9,2,6,5,3,5,8,9,7,9,3) \mapsto$
$(1,3,1,4,9,5,6,2,3,5,8,5,7,9,3,9)$.

NOT_1 gate on 3 qubits:
$(3, 1, 4, 1, 5, 9, 2, 6) \mapsto$
$(4, 1, 3, 1, 2, 6, 5, 9)$.

NOT gates

NOT_0 gate on 3 qubits:
$(3, 1, 4, 1, 5, 9, 2, 6) \mapsto$
$(1, 3, 1, 4, 9, 5, 6, 2)$.

NOT_0 gate on 4 qubits:
$(3,1,4,1,5,9,2,6,5,3,5,8,9,7,9,3) \mapsto$
$(1,3,1,4,9,5,6,2,3,5,8,5,7,9,3,9)$.

NOT_1 gate on 3 qubits:
$(3, 1, 4, 1, 5, 9, 2, 6) \mapsto$
$(4, 1, 3, 1, 2, 6, 5, 9)$.

NOT_2 gate on 3 qubits:
$(3, 1, 4, 1, 5, 9, 2, 6) \mapsto$
$(5, 9, 2, 6, 3, 1, 4, 1)$.

state	measurement
$(1,0,0,0,0,0,0,0)$	000
$(0,1,0,0,0,0,0,0)$	001
$(0,0,1,0,0,0,0,0)$	010
$(0,0,0,1,0,0,0,0)$	011
$(0,0,0,0,1,0,0,0)$	100
$(0,0,0,0,0,1,0,0)$	101
$(0,0,0,0,0,0,1,0)$	110
$(0,0,0,0,0,0,0,1)$	111

Operation on quantum state:
NOT_0, swapping pairs.

Operation after measurement:
flipping bit 0 of result.

Flip: output is not input.

Controlled-NOT gates

e.g. $\text{CNOT}_{1,0}$:

$(3, 1, 4, 1, 5, 9, 2, 6) \mapsto$

$(3, 1, 1, 4, 5, 9, 6, 2)$.

Controlled-NOT gates

e.g. $\mathrm{CNOT}_{1,0}$:
$(3, 1, 4, 1, 5, 9, 2, 6) \mapsto$
$(3, 1, 1, 4, 5, 9, 6, 2)$.

Operation after measurement:
flipping bit 0 *if* bit 1 is set; i.e.,
$(q_2, q_1, q_0) \mapsto (q_2, q_1, q_0 \oplus q_1)$.

Controlled-NOT gates

e.g. $\text{CNOT}_{1,0}$:
$(3, 1, 4, 1, 5, 9, 2, 6) \mapsto$
$(3, 1, 1, 4, 5, 9, 6, 2)$.

Operation after measurement:
flipping bit 0 *if* bit 1 is set; i.e.,
$(q_2, q_1, q_0) \mapsto (q_2, q_1, q_0 \oplus q_1)$.

e.g. $\text{CNOT}_{2,0}$:
$(3, 1, 4, 1, 5, 9, 2, 6) \mapsto$
$(3, 1, 4, 1, 9, 5, 6, 2)$.

Controlled-NOT gates

e.g. $\text{CNOT}_{1,0}$:
$(3, 1, 4, 1, 5, 9, 2, 6) \mapsto$
$(3, 1, 1, 4, 5, 9, 6, 2)$.

Operation after measurement:
flipping bit 0 *if* bit 1 is set; i.e.,
$(q_2, q_1, q_0) \mapsto (q_2, q_1, q_0 \oplus q_1)$.

e.g. $\text{CNOT}_{2,0}$:
$(3, 1, 4, 1, 5, 9, 2, 6) \mapsto$
$(3, 1, 4, 1, 9, 5, 6, 2)$.

e.g. $\text{CNOT}_{0,2}$:
$(3, 1, 4, 1, 5, 9, 2, 6) \mapsto$
$(3, 9, 4, 6, 5, 1, 2, 1)$.

Toffoli gates

Also known as controlled-controlled-NOT gates.

e.g. $\text{CCNOT}_{2,1,0}$:
$(3, 1, 4, 1, 5, 9, 2, 6) \mapsto$
$(3, 1, 4, 1, 5, 9, 6, 2)$.

Toffoli gates

Also known as controlled-controlled-NOT gates.

e.g. $\text{CCNOT}_{2,1,0}$:
$(3, 1, 4, 1, 5, 9, 2, 6) \mapsto$
$(3, 1, 4, 1, 5, 9, 6, 2)$.

Operation after measurement:
$(q_2, q_1, q_0) \mapsto (q_2, q_1, q_0 \oplus q_1 q_2)$.

Toffoli gates

Also known as
controlled-controlled-NOT gates.

e.g. $\text{CCNOT}_{2,1,0}$:
$(3, 1, 4, 1, 5, 9, 2, 6) \mapsto$
$(3, 1, 4, 1, 5, 9, 6, 2)$.

Operation after measurement:
$(q_2, q_1, q_0) \mapsto (q_2, q_1, q_0 \oplus q_1 q_2)$.

e.g. $\text{CCNOT}_{0,1,2}$:
$(3, 1, 4, 1, 5, 9, 2, 6) \mapsto$
$(3, 1, 4, 6, 5, 9, 2, 1)$.

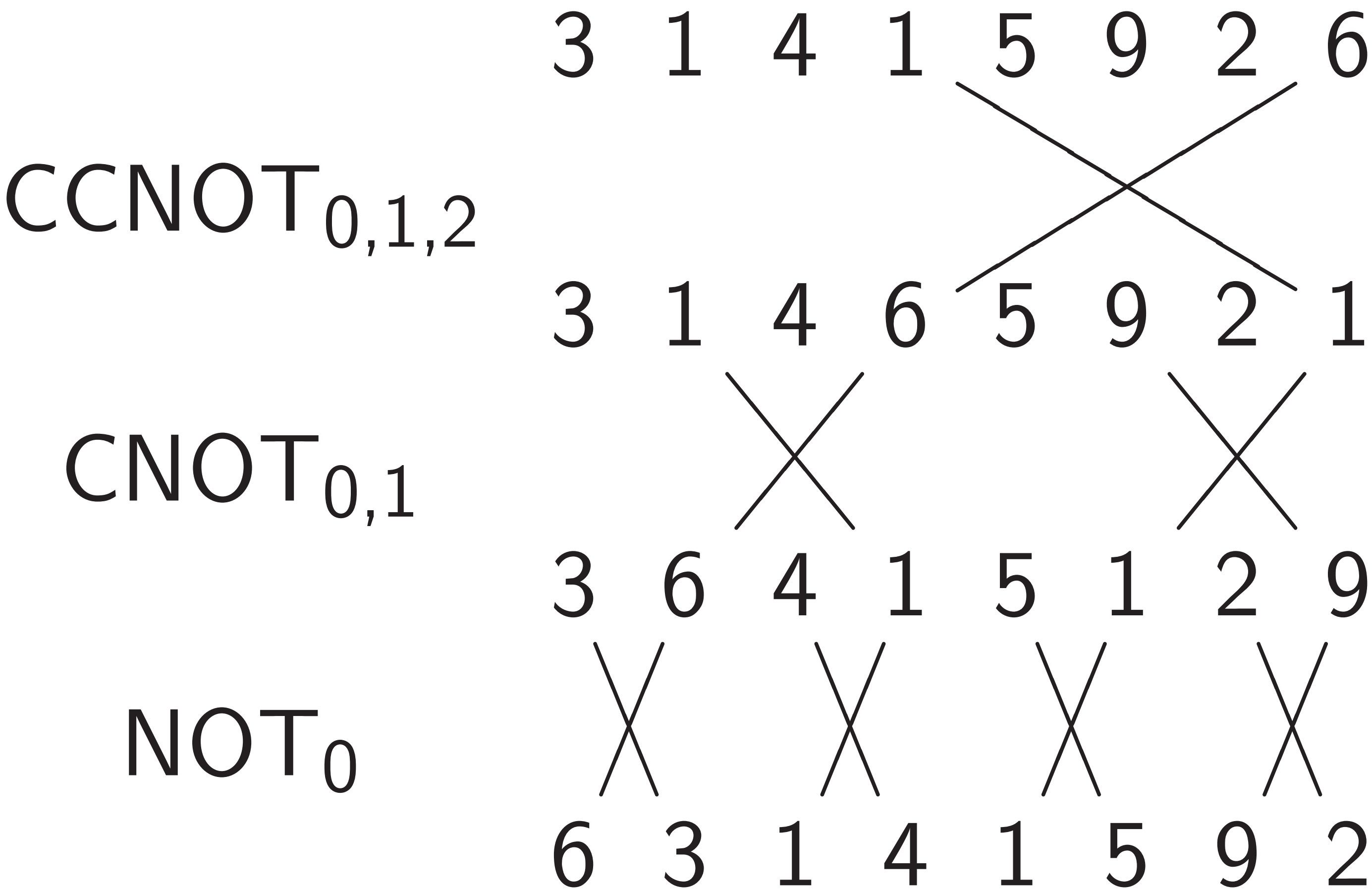
More shuffling

Combine NOT, CNOT, Toffoli to build other permutations.

More shuffling

Combine NOT, CNOT, Toffoli to build other permutations.

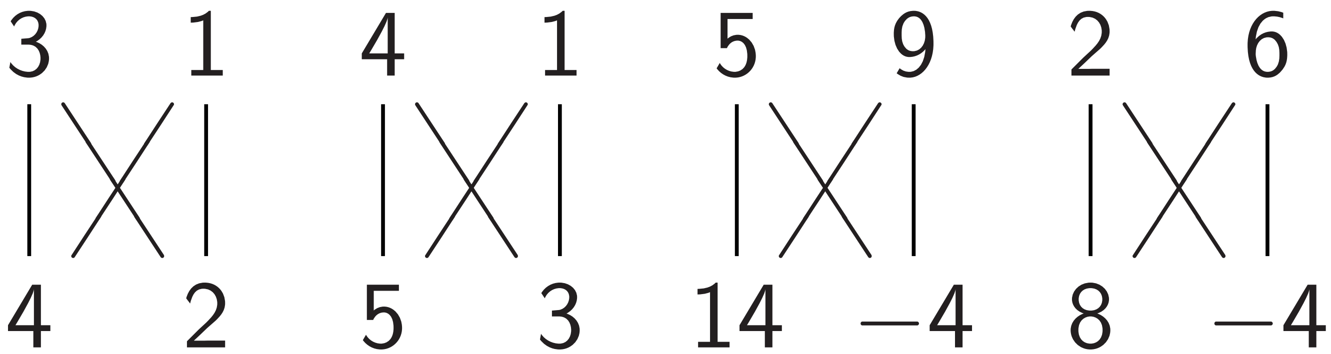
e.g. series of gates to rotate 8 positions by distance 1:



Hadamard gates

Hadamard$_0$:

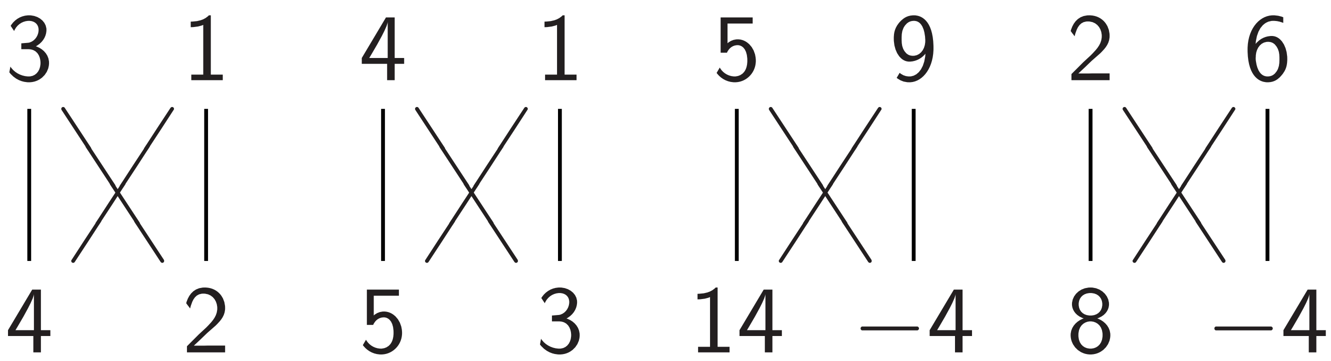
$(a, b) \mapsto (a + b, a - b)$.



Hadamard gates

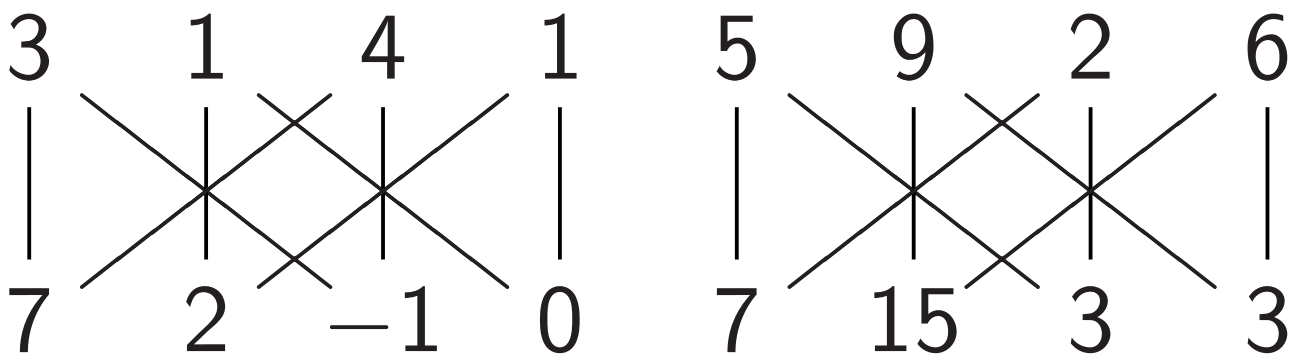
Hadamard$_0$:

$(a, b) \mapsto (a + b, a - b)$.



Hadamard$_1$:

$(a, b, c, d) \mapsto$
$(a + c, b + d, a - c, b - d)$.



Simon's algorithm

Step 1. Set up pure zero state:

1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0.

Simon's algorithm

Step 2. Hadamard$_0$:

1, 1, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0.

Simon's algorithm

Step 3. Hadamard$_1$:

1, 1, 1, 1, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0.

Simon's algorithm

Step 4. Hadamard_2:

1, 1, 1, 1, 1, 1, 1, 1,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0.

Each column is a parallel universe.

Simon's algorithm

Step 5. $\text{CNOT}_{0,3}$:

1, 0, 1, 0, 1, 0, 1, 0,

0, 1, 0, 1, 0, 1, 0, 1,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0.

Each column is a parallel universe performing its own computations.

Simon's algorithm

Step 5b. More shuffling:

1, 0, 0, 0, 1, 0, 0, 0,

0, 1, 0, 0, 0, 1, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 1, 0, 0, 0, 1, 0,

0, 0, 0, 1, 0, 0, 0, 1,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0.

Each column is a parallel universe performing its own computations.

Simon's algorithm

Step 5c. More shuffling:

1, 0, 0, 0, 0, 0, 0, 0,

0, 1, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 1, 0, 0, 0,

0, 0, 0, 0, 0, 1, 0, 0,

0, 0, 1, 0, 0, 0, 0, 0,

0, 0, 0, 1, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 1, 0,

0, 0, 0, 0, 0, 0, 0, 1.

Each column is a parallel universe performing its own computations.

Simon's algorithm

Step 5d. More shuffling:

1, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 1, 0, 0,

0, 0, 0, 0, 1, 0, 0, 0,

0, 1, 0, 0, 0, 0, 0, 0,

0, 0, 1, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 1,

0, 0, 0, 0, 0, 0, 1, 0,

0, 0, 0, 1, 0, 0, 0, 0.

Each column is a parallel universe performing its own computations.

Simon's algorithm

Step 5e. More shuffling:

1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0.

Each column is a parallel universe performing its own computations.

Simon's algorithm

Step 5f. More shuffling:

0, 0, 0, 0, 0, 1, 0, 0,

1, 0, 0, 0, 0, 0, 0, 0,

0, 1, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 1, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 1, 0, 0, 0, 0, 1,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 1, 0, 0, 1, 0.

Each column is a parallel universe performing its own computations.

Simon's algorithm

Step 5g. More shuffling:

0, 1, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 1, 0, 0, 0,

0, 0, 0, 0, 0, 1, 0, 0,

1, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 1, 0, 0, 1, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 1, 0, 0, 0, 0, 1.

Each column is a parallel universe performing its own computations.

Simon's algorithm

Step 5h. More shuffling:

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 1, 0, 0, 1, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 1, 0, 0, 0, 0, 1,

0, 1, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 1, 0, 0, 0,

0, 0, 0, 0, 0, 1, 0, 0,

1, 0, 0, 0, 0, 0, 0, 0.

Each column is a parallel universe performing its own computations.

Simon's algorithm

Step 5i. More shuffling:

0, 0, 0, 0, 0, 0, 1, 0,

0, 0, 0, 1, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 1,

0, 0, 1, 0, 0, 0, 0, 0,

0, 1, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 1, 0, 0, 0,

0, 0, 0, 0, 0, 1, 0, 0,

1, 0, 0, 0, 0, 0, 0, 0.

Each column is a parallel universe performing its own computations.

Simon's algorithm

Step 5j. Final shuffling:

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 1, 0, 0, 1, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 1, 0, 0, 0, 0, 1,

0, 1, 0, 0, 1, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

1, 0, 0, 0, 0, 1, 0, 0.

Each column is a parallel universe performing its own computations.

Simon's algorithm

Step 5j. Final shuffling:

0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 1,
0, 1, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 1, 0, 0.

Each column is a parallel universe performing its own computations.

Surprise: u and $u \oplus 101$ match.

Simon's algorithm

Step 6. Hadamard$_0$:

$0, 0, 0, 0, 0, 0, 0, 0,$

$0, 0, 1, \overline{1}, 0, 0, 1, 1,$

$0, 0, 0, 0, 0, 0, 0, 0,$

$0, 0, 1, 1, 0, 0, 1, \overline{1},$

$1, \overline{1}, 0, 0, 1, 1, 0, 0,$

$0, 0, 0, 0, 0, 0, 0, 0,$

$0, 0, 0, 0, 0, 0, 0, 0,$

$1, 1, 0, 0, 1, \overline{1}, 0, 0.$

Simon's algorithm

Step 7. Hadamard$_1$:

$0, 0, 0, 0, 0, 0, 0, 0,$

$1, \overline{1}, \overline{1}, 1, 1, 1, \overline{1}, \overline{1},$

$0, 0, 0, 0, 0, 0, 0, 0,$

$1, 1, \overline{1}, \overline{1}, 1, \overline{1}, \overline{1}, 1,$

$1, \overline{1}, 1, \overline{1}, 1, 1, 1, 1,$

$0, 0, 0, 0, 0, 0, 0, 0,$

$0, 0, 0, 0, 0, 0, 0, 0,$

$1, 1, 1, 1, 1, \overline{1}, 1, \overline{1}.$

Simon's algorithm

Step 8. Hadamard$_2$:

0, 0, 0, 0, 0, 0, 0, 0,

2, 0, $\overline{2}$, 0, 0, $\overline{2}$, 0, 2,

0, 0, 0, 0, 0, 0, 0, 0,

2, 0, $\overline{2}$, 0, 0, 2, 0, $\overline{2}$,

2, 0, 2, 0, 0, $\overline{2}$, 0, $\overline{2}$,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

2, 0, 2, 0, 0, 2, 0, 2.

Simon's algorithm

Step 8. Hadamard$_2$:

0, 0, 0, 0, 0, 0, 0, 0,

2, 0, $\overline{2}$, 0, 0, $\overline{2}$, 0, 2,

0, 0, 0, 0, 0, 0, 0, 0,

2, 0, $\overline{2}$, 0, 0, 2, 0, $\overline{2}$,

2, 0, 2, 0, 0, $\overline{2}$, 0, $\overline{2}$,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

2, 0, 2, 0, 0, 2, 0, 2.

Step 9: Measure. Obtain some information about the surprise: a random vector orthogonal to 101.