Quantum algorithms for the subset-sum problem

D. J. Bernstein
University of Illinois at Chicago &
Technische Universiteit Eindhoven

cr.yp.to/qsubsetsum.html

Joint work with:

Stacey Jeffery
University of Waterloo

Tanja Lange
Technische Universiteit Eindhoven

Alexander Meurer
Ruhr-Universität Bochum

Subset-sum example:
Is there a subsequence of (499, 852, 1927, 2535, 3596, 3608, 4688, 5989, 6385, 7353, 7650, 9413) having sum 36634?

Many variations: e.g.,
find such a subsequence *if* one exists;
find such a subsequence *knowing that* one exists;
allow range of sums;
coefficients outside $\{0, 1\}$; etc.

"Subset-sum problem"; "knapsack problem"; etc.

m algorithms
subset-sum problem

ernstein
ty of Illinois at Chicago &
che Universiteit Eindhoven

to/qsubsetsum.html

rk with:

Jeffery
ty of Waterloo

ange
che Universiteit Eindhoven

er Meurer
iversität Bochum

Subset-sum example:
Is there a subsequence of
(499, 852, 1927, 2535, 3596, 3608,
4688, 5989, 6385, 7353, 7650, 9413)
having sum 36634?

Many variations: e.g.,
find such a subsequence
if one exists;
find such a subsequence
knowing that one exists;
allow range of sums;
coefficients outside $\{0, 1\}$; etc.

"Subset-sum problem";
"knapsack problem"; etc.

The latt

Define x

Define L
$\{v : v_1 x$

Define u
$(70, 2, 0,$

If $J \subseteq \{$
and $\sum_{i \in}$
$v \in L$ w

v is very
Reasona
v is the
Subset-s
codimen

ms

n problem

is at Chicago &

siteit Eindhoven

etsum.html

erloo

siteit Eindhoven

Bochum

Subset-sum example:

Is there a subsequence of
(499, 852, 1927, 2535, 3596, 3608,
4688, 5989, 6385, 7353, 7650, 9413)
having sum 36634?

Many variations: e.g.,
find such a subsequence
if one exists;
find such a subsequence
knowing that one exists;
allow range of sums;
coefficients outside $\{0, 1\}$; etc.

"Subset-sum problem";
"knapsack problem"; etc.

The lattice connec

Define $x_1 = 499$,

Define $L \subseteq \mathbf{Z}^{12}$ as
$\{v : v_1 x_1 + \cdots +$

Define $u \in \mathbf{Z}^{12}$ as
$(70, 2, 0, 0, 0, 0, 0,$

If $J \subseteq \{1, 2, \ldots, 1$
and $\sum_{i \in J} x_i = 36$
$v \in L$ where $v_i =$

v is very close to
Reasonable to hop
v is the closest ve
Subset-sum algorit
codimension-1 CV

ago &
hoven

ml

hoven

Subset-sum example:

Is there a subsequence of
(499, 852, 1927, 2535, 3596, 3608,
4688, 5989, 6385, 7353, 7650, 9413)
having sum 36634?

Many variations: e.g.,
find such a subsequence
if one exists;
find such a subsequence
knowing that one exists;
allow range of sums;
coefficients outside $\{0, 1\}$; etc.

"Subset-sum problem";
"knapsack problem"; etc.

<u>The lattice connection</u>

Define $x_1 = 499, \ldots, x_{12} =$

Define $L \subseteq \mathbf{Z}^{12}$ as
$\{v : v_1 x_1 + \cdots + v_{12} x_{12} =$

Define $u \in \mathbf{Z}^{12}$ as
(70, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

If $J \subseteq \{1, 2, \ldots, 12\}$
and $\sum_{i \in J} x_i = 36634$ then
$v \in L$ where $v_i = u_i - [i \in$

v is very close to u.
Reasonable to hope that
v is the closest vector in L t
Subset-sum algorithms $\approx$
codimension-1 CVP algorith

Subset-sum example:

Is there a subsequence of (499, 852, 1927, 2535, 3596, 3608, 4688, 5989, 6385, 7353, 7650, 9413) having sum 36634?

Many variations: e.g.,
find such a subsequence *if* one exists;
find such a subsequence *knowing that* one exists;
allow range of sums;
coefficients outside $\{0, 1\}$; etc.

"Subset-sum problem"; "knapsack problem"; etc.

The lattice connection

Define $x_1 = 499, \ldots, x_{12} = 9413$.

Define $L \subseteq \mathbf{Z}^{12}$ as $\{v : v_1 x_1 + \cdots + v_{12} x_{12} = 0\}$.

Define $u \in \mathbf{Z}^{12}$ as $(70, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$.

If $J \subseteq \{1, 2, \ldots, 12\}$ and $\sum_{i \in J} x_i = 36634$ then $v \in L$ where $v_i = u_i - [i \in J]$.

v is very close to u.
Reasonable to hope that v is the closest vector in L to u.
Subset-sum algorithms $\approx$ codimension-1 CVP algorithms.

sum example:
a subsequence of
2, 1927, 2535, 3596, 3608,
89, 6385, 7353, 7650, 9413)
um 36634?

ariations: e.g.,
n a subsequence
xists;
n a subsequence
that one exists;
nge of sums;
nts outside $\{0,1\}$; etc.

-sum problem";
ck problem"; etc.

The lattice connection

Define $x_1 = 499, \ldots, x_{12} = 9413$.

Define $L \subseteq \mathbf{Z}^{12}$ as
$\{v : v_1 x_1 + \cdots + v_{12} x_{12} = 0\}$.

Define $u \in \mathbf{Z}^{12}$ as
$(70, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$.

If $J \subseteq \{1, 2, \ldots, 12\}$
and $\sum_{i \in J} x_i = 36634$ then
$v \in L$ where $v_i = u_i - [i \in J]$.

v is very close to u.
Reasonable to hope that
v is the closest vector in L to u.
Subset-sum algorithms $\approx$
codimension-1 CVP algorithms.

The cod

A weigh
Is there
(499, 85
4688, 59
having l

ple:
ence of
535, 3596, 3608,
7353, 7650, 9413)
?

e.g.,
quence

quence
exists;
ns;
e $\{0, 1\}$; etc.

lem";
n"; etc.

The lattice connection

Define $x_1 = 499$, ..., $x_{12} = 9413$.

Define $L \subseteq \mathbf{Z}^{12}$ as
$\{v : v_1 x_1 + \cdots + v_{12} x_{12} = 0\}$.

Define $u \in \mathbf{Z}^{12}$ as
$(70, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$.

If $J \subseteq \{1, 2, \ldots, 12\}$
and $\sum_{i \in J} x_i = 36634$ then
$v \in L$ where $v_i = u_i - [i \in J]$.

v is very close to u.
Reasonable to hope that
v is the closest vector in L to u.
Subset-sum algorithms $\approx$
codimension-1 CVP algorithms.

The coding connec

A weight-w subset
Is there a subsequ
(499, 852, 1927, 25
4688, 5989, 6385, 7
having length w a

3608,
, 9413)

tc.

The lattice connection

Define $x_1 = 499, \ldots, x_{12} = 9413$.

Define $L \subseteq \mathbf{Z}^{12}$ as
$\{v : v_1 x_1 + \cdots + v_{12} x_{12} = 0\}$.

Define $u \in \mathbf{Z}^{12}$ as
$(70, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$.

If $J \subseteq \{1, 2, \ldots, 12\}$
and $\sum_{i \in J} x_i = 36634$ then
$v \in L$ where $v_i = u_i - [i \in J]$.

v is very close to u.
Reasonable to hope that
v is the closest vector in L to u.
Subset-sum algorithms $\approx$
codimension-1 CVP algorithms.

The coding connection

A weight-w subset-sum prob
Is there a subsequence of
(499, 852, 1927, 2535, 3596,
4688, 5989, 6385, 7353, 7650
having length w and sum 36

The lattice connection

Define $x_1 = 499, \ldots, x_{12} = 9413$.

Define $L \subseteq \mathbf{Z}^{12}$ as
$\{v : v_1x_1 + \cdots + v_{12}x_{12} = 0\}$.

Define $u \in \mathbf{Z}^{12}$ as
$(70, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$.

If $J \subseteq \{1, 2, \ldots, 12\}$
and $\sum_{i \in J} x_i = 36634$ then
$v \in L$ where $v_i = u_i - [i \in J]$.

v is very close to u.
Reasonable to hope that
v is the closest vector in L to u.
Subset-sum algorithms $\approx$
codimension-1 CVP algorithms.

The coding connection

A weight-w subset-sum problem:
Is there a subsequence of
$(499, 852, 1927, 2535, 3596, 3608, 4688, 5989, 6385, 7353, 7650, 9413)$
having length w and sum 36634?

The lattice connection

Define $x_1 = 499, \ldots, x_{12} = 9413$.

Define $L \subseteq \mathbf{Z}^{12}$ as $\{v : v_1 x_1 + \cdots + v_{12} x_{12} = 0\}$.

Define $u \in \mathbf{Z}^{12}$ as $(70, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$.

If $J \subseteq \{1, 2, \ldots, 12\}$ and $\sum_{i \in J} x_i = 36634$ then $v \in L$ where $v_i = u_i - [i \in J]$.

v is very close to u. Reasonable to hope that v is the closest vector in L to u. Subset-sum algorithms $\approx$ codimension-1 CVP algorithms.

The coding connection

A weight-w subset-sum problem: Is there a subsequence of $(499, 852, 1927, 2535, 3596, 3608, 4688, 5989, 6385, 7353, 7650, 9413)$ having length w and sum 36634?

Replace $\mathbf{Z}$ with $(\mathbf{Z}/2)^m$: Is there a subsequence of $(499, 852, 1927, 2535, 3596, 3608, 4688, 5989, 6385, 7353, 7650, 9413)$ having length w and xor 1060?

This is the central algorithmic problem in coding theory.

ice connection

$_1 = 499, \ldots, x_{12} = 9413$.

$\subseteq \mathbf{Z}^{12}$ as

$_1 + \cdots + v_{12}x_{12} = 0\}$.

$u \in \mathbf{Z}^{12}$ as

$, 0, 0, 0, 0, 0, 0, 0, 0, 0)$.

$1, 2, \ldots, 12\}$

$_{\in J} x_i = 36634$ then

here $v_i = u_i - [i \in J]$.

close to u.

ble to hope that

closest vector in L to u.

sum algorithms $\approx$

sion-1 CVP algorithms.

The coding connection

A weight-w subset-sum problem:
Is there a subsequence of
$(499, 852, 1927, 2535, 3596, 3608,$
$4688, 5989, 6385, 7353, 7650, 9413)$
having length w and sum 36634?

Replace $\mathbf{Z}$ with $(\mathbf{Z}/2)^m$:
Is there a subsequence of
$(499, 852, 1927, 2535, 3596, 3608,$
$4688, 5989, 6385, 7353, 7650, 9413)$
having length w and xor 1060?

This is the central algorithmic problem in coding theory.

Recent a

Eurocryp
Howgrav
subset-s
(Incorrec

Eurocryp
Becker–
subset-s

Adaptat
Asiacryp
Thomae
Becker–J

ction

$\ldots, x_{12} = 9413$.

g

$v_{12}x_{12} = 0\}$.

$0, 0, 0, 0, 0)$.

$2\}$

634 then

$u_i - [i \in J]$.

u.

e that

ctor in L to u.

thms $\approx$

P algorithms.

The coding connection

A weight-w subset-sum problem:
Is there a subsequence of
$(499, 852, 1927, 2535, 3596, 3608,$
$4688, 5989, 6385, 7353, 7650, 9413)$
having length w and sum 36634?

Replace $\mathbf{Z}$ with $(\mathbf{Z}/2)^m$:
Is there a subsequence of
$(499, 852, 1927, 2535, 3596, 3608,$
$4688, 5989, 6385, 7353, 7650, 9413)$
having length w and xor 1060?

This is the central algorithmic problem in coding theory.

Recent asymptotic

Eurocrypt 2010
Howgrave-Graham
subset-sum expone
(Incorrect claim:

Eurocrypt 2011
Becker–Coron–Jou
subset-sum expone

Adaptations to de
Asiacrypt 2011 Ma
Thomae, Eurocryp
Becker–Joux–May

= 9413.

0}.

).

J].

to u.

ms.

The coding connection

A weight-w subset-sum problem:
Is there a subsequence of
$(499, 852, 1927, 2535, 3596, 3608, 4688, 5989, 6385, 7353, 7650, 9413)$
having length w and sum 36634?

Replace $\mathbf{Z}$ with $(\mathbf{Z}/2)^m$:
Is there a subsequence of
$(499, 852, 1927, 2535, 3596, 3608, 4688, 5989, 6385, 7353, 7650, 9413)$
having length w and xor 1060?

This is the central algorithmic problem in coding theory.

Recent asymptotic news

Eurocrypt 2010
Howgrave-Graham–Joux:
subset-sum exponent ≈0.33
(Incorrect claim: ≈0.311.)

Eurocrypt 2011
Becker–Coron–Joux:
subset-sum exponent ≈0.29

Adaptations to decoding:
Asiacrypt 2011 May–Meurer
Thomae, Eurocrypt 2012
Becker–Joux–May–Meurer.

The coding connection

A weight-w subset-sum problem:
Is there a subsequence of
$(499, 852, 1927, 2535, 3596, 3608, 4688, 5989, 6385, 7353, 7650, 9413)$
having length w and sum 36634?

Replace $\mathbf{Z}$ with $(\mathbf{Z}/2)^m$:
Is there a subsequence of
$(499, 852, 1927, 2535, 3596, 3608, 4688, 5989, 6385, 7353, 7650, 9413)$
having length w and xor 1060?

This is the central algorithmic problem in coding theory.

Recent asymptotic news

Eurocrypt 2010
Howgrave-Graham–Joux:
subset-sum exponent ≈0.337.
(Incorrect claim: ≈0.311.)

Eurocrypt 2011
Becker–Coron–Joux:
subset-sum exponent ≈0.291.

Adaptations to decoding:
Asiacrypt 2011 May–Meurer–Thomae, Eurocrypt 2012 Becker–Joux–May–Meurer.

ing connection

t-w subset-sum problem:
a subsequence of
2, 1927, 2535, 3596, 3608,
89, 6385, 7353, 7650, 9413)
ength w and sum 36634?

Z with $(\mathbf{Z}/2)^m$:
a subsequence of
2, 1927, 2535, 3596, 3608,
89, 6385, 7353, 7650, 9413)
ength w and xor 1060?

he central algorithmic
in coding theory.

Recent asymptotic news

Eurocrypt 2010
Howgrave-Graham–Joux:
subset-sum exponent ≈0.337.
(Incorrect claim: ≈0.311.)

Eurocrypt 2011
Becker–Coron–Joux:
subset-sum exponent ≈0.291.

Adaptations to decoding:
Asiacrypt 2011 May–Meurer–
Thomae, Eurocrypt 2012
Becker–Joux–May–Meurer.

Post-qua

Claimed
Lyubash
"Public-
primitive
as secur

There ar
quantum
better th
on the s

Hmmm.
quantum

ction

t-sum problem:
ence of
535, 3596, 3608,
7353, 7650, 9413)
nd sum 36634?

Z/2)^m:
ence of
535, 3596, 3608,
7353, 7650, 9413)
nd xor 1060?

algorithmic
theory.

Recent asymptotic news

Eurocrypt 2010 Howgrave-Graham–Joux: subset-sum exponent ≈0.337. (Incorrect claim: ≈0.311.)

Eurocrypt 2011 Becker–Coron–Joux: subset-sum exponent ≈0.291.

Adaptations to decoding: Asiacrypt 2011 May–Meurer–Thomae, Eurocrypt 2012 Becker–Joux–May–Meurer.

Post-quantum sub

Claimed in TCC 2
Lyubashevsky–Pal
"Public-key crypto
primitives provably
as secure as subse

There are "current
quantum algorithm
better than classic
on the subset sum

Hmmm. What's t
quantum subset-s

blem:

3608,

, 9413)

5634?

3608,

, 9413)

60?

nic

Recent asymptotic news

Eurocrypt 2010
Howgrave-Graham–Joux:
subset-sum exponent ≈0.337.
(Incorrect claim: ≈0.311.)

Eurocrypt 2011
Becker–Coron–Joux:
subset-sum exponent ≈0.291.

Adaptations to decoding:
Asiacrypt 2011 May–Meurer–Thomae, Eurocrypt 2012 Becker–Joux–May–Meurer.

Post-quantum subset sum

Claimed in TCC 2010
Lyubashevsky–Palacio–Segev
"Public-key cryptographic primitives provably as secure as subset sum":

There are "currently no know
quantum algorithms that pe
better than classical ones
on the subset sum problem"

Hmmm. What's the best
quantum subset-sum expone

Recent asymptotic news

Eurocrypt 2010
Howgrave-Graham–Joux:
subset-sum exponent ≈0.337.
(Incorrect claim: ≈0.311.)

Eurocrypt 2011
Becker–Coron–Joux:
subset-sum exponent ≈0.291.

Adaptations to decoding:
Asiacrypt 2011 May–Meurer–Thomae, Eurocrypt 2012 Becker–Joux–May–Meurer.

Post-quantum subset sum

Claimed in TCC 2010
Lyubashevsky–Palacio–Segev
"Public-key cryptographic primitives provably as secure as subset sum":

There are "currently no known quantum algorithms that perform better than classical ones on the subset sum problem".

Hmmm. What's the best *quantum* subset-sum exponent?

asymptotic news

pt 2010
ve-Graham–Joux:
um exponent ≈0.337.
ct claim: ≈0.311.)

pt 2011
Coron–Joux:
um exponent ≈0.291.

ions to decoding:
pt 2011 May–Meurer–
, Eurocrypt 2012
Joux–May–Meurer.

Post-quantum subset sum

Claimed in TCC 2010 Lyubashevsky–Palacio–Segev "Public-key cryptographic primitives provably as secure as subset sum":

There are "currently no known quantum algorithms that perform better than classical ones on the subset sum problem".

Hmmm. What's the best *quantum* subset-sum exponent?

Interlude

Textboo

Proof o

New

Proof

Mislead
that bes
best *pro*

c news

n–Joux:
ent ≈0.337.
≈0.311.)

ux:
ent ≈0.291.

coding:
ay–Meurer–
ot 2012
–Meurer.

Post-quantum subset sum

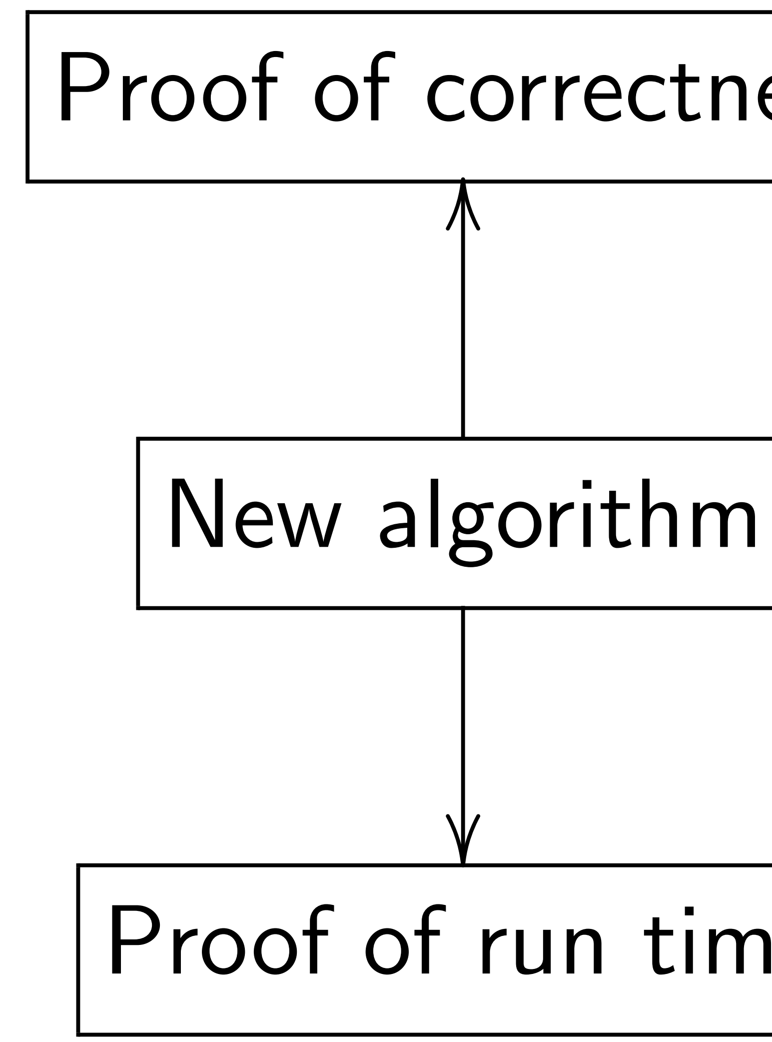
Claimed in TCC 2010 Lyubashevsky–Palacio–Segev "Public-key cryptographic primitives provably as secure as subset sum":

There are "currently no known quantum algorithms that perform better than classical ones on the subset sum problem".

Hmmm. What's the best *quantum* subset-sum exponent?

Interlude: Algorith

Textbook algorith



Mislead students i
that best algorith
best *proven* algorit

7.

1.

–

Post-quantum subset sum

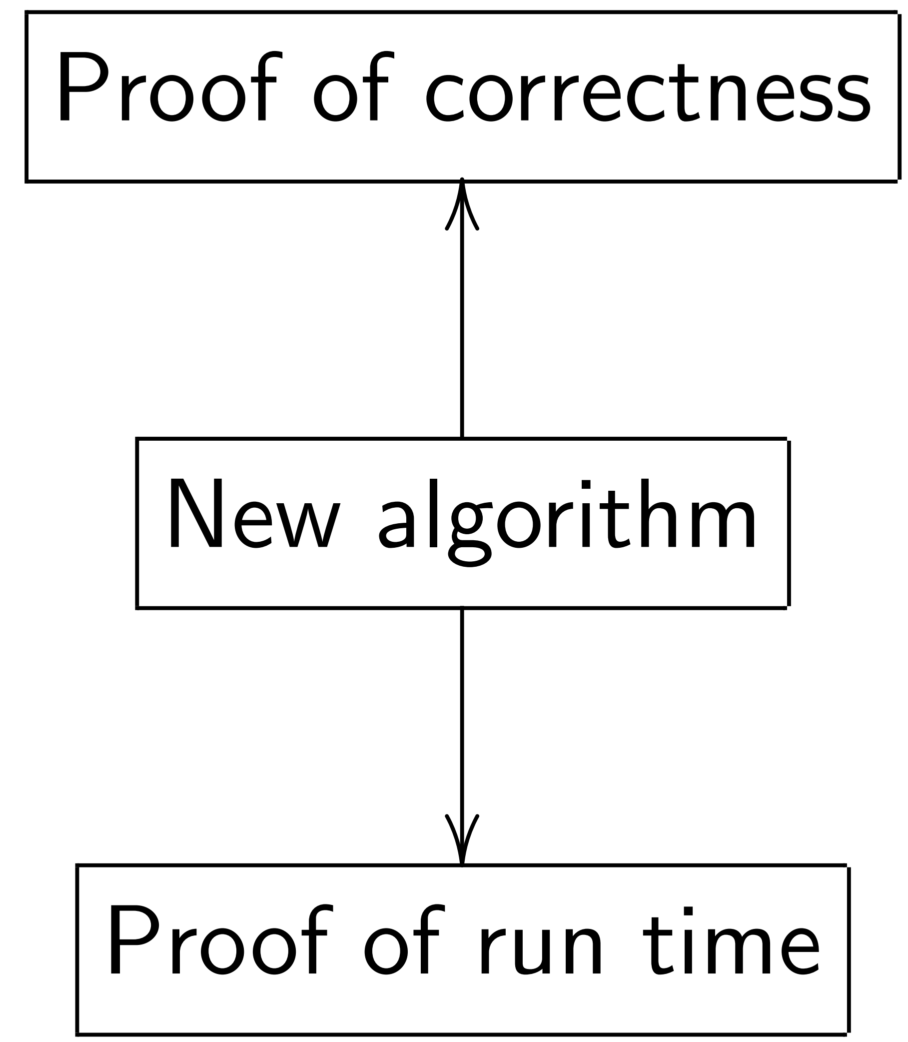
Claimed in TCC 2010 Lyubashevsky–Palacio–Segev "Public-key cryptographic primitives provably as secure as subset sum":

There are "currently no known quantum algorithms that perform better than classical ones on the subset sum problem".

Hmmm. What's the best *quantum* subset-sum exponent?

Interlude: Algorithm design

Textbook algorithm analysis



Mislead students into thinki
that best algorithm =
best *proven* algorithm.

Post-quantum subset sum

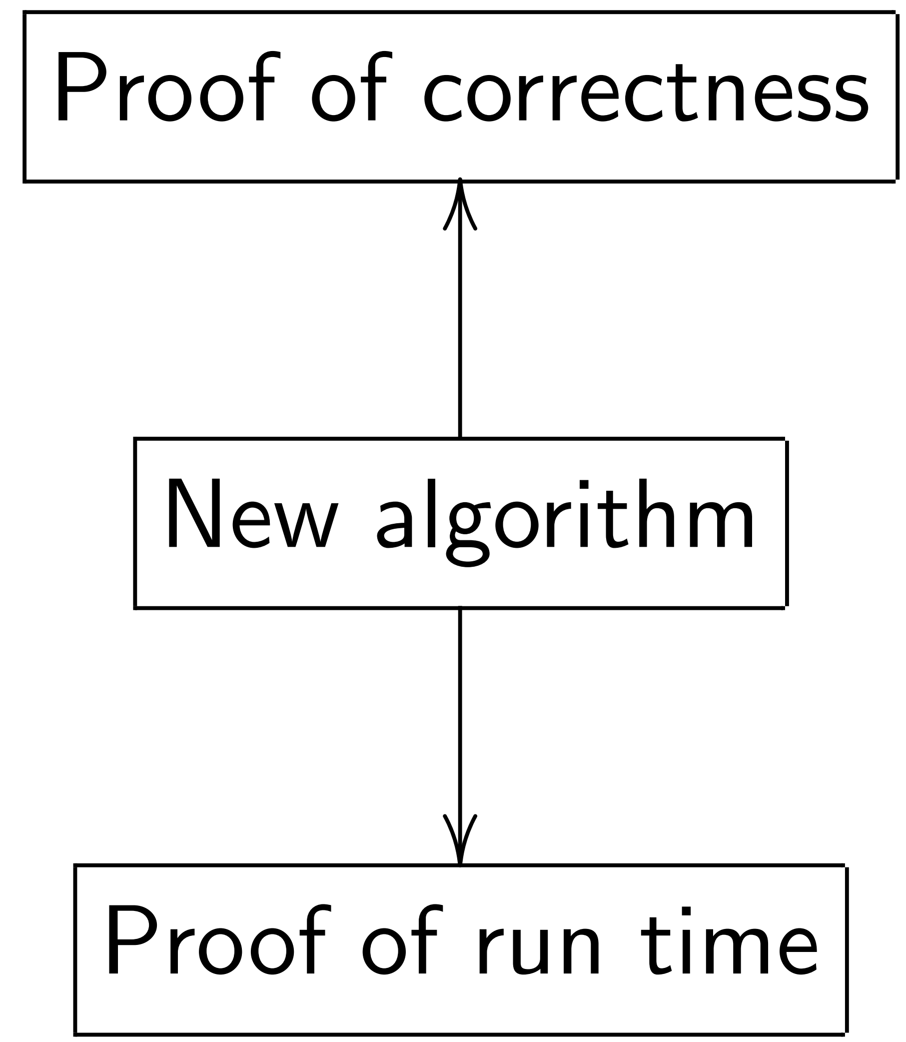
Claimed in TCC 2010 Lyubashevsky–Palacio–Segev "Public-key cryptographic primitives provably as secure as subset sum":

There are "currently no known quantum algorithms that perform better than classical ones on the subset sum problem".

Hmmm. What's the best *quantum* subset-sum exponent?

Interlude: Algorithm design

Textbook algorithm analysis:



Mislead students into thinking that best algorithm = best *proven* algorithm.

antum subset sum

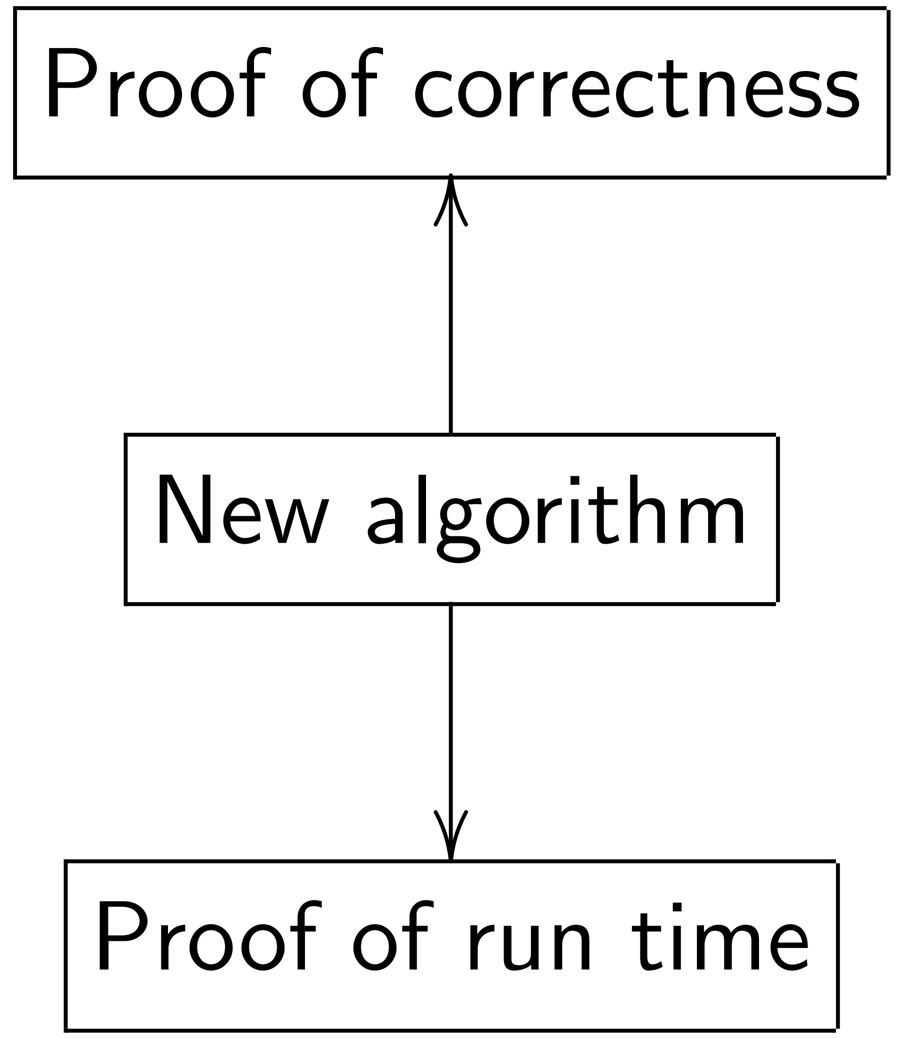
in TCC 2010
evsky–Palacio–Segev
key cryptographic
es provably
e as subset sum":

re "currently no known
n algorithms that perform
han classical ones
ubset sum problem".

What's the best
n subset-sum exponent?

Interlude: Algorithm design

Textbook algorithm analysis:



Mislead students into thinking that best algorithm = best *proven* algorithm.

Reality:
cryptana
are almo

oset sum

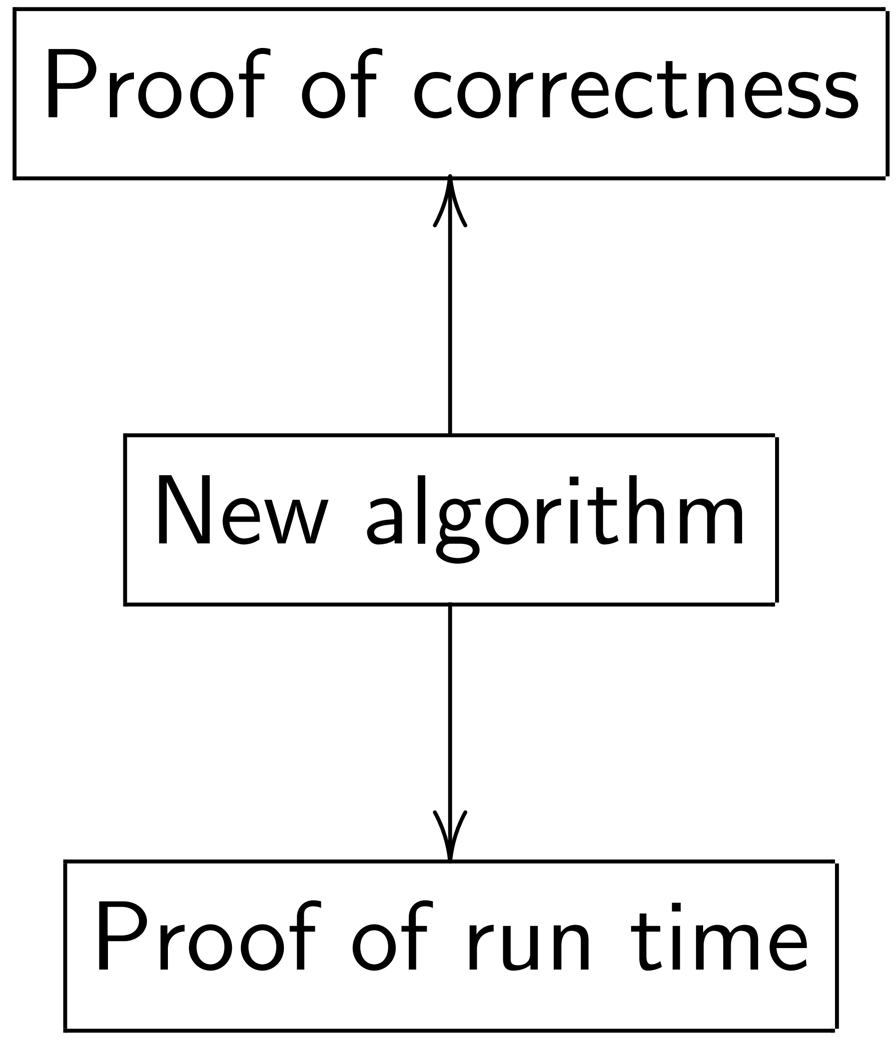
010
acio–Segev
ographic
y
t sum":

tly no known
ns that perform
al ones
problem".

he best
um exponent?

Interlude: Algorithm design

Textbook algorithm analysis:



Mislead students into thinking that best algorithm = best *proven* algorithm.

Reality: state-of-th
cryptanalytic algor
are almost never p

v

wn

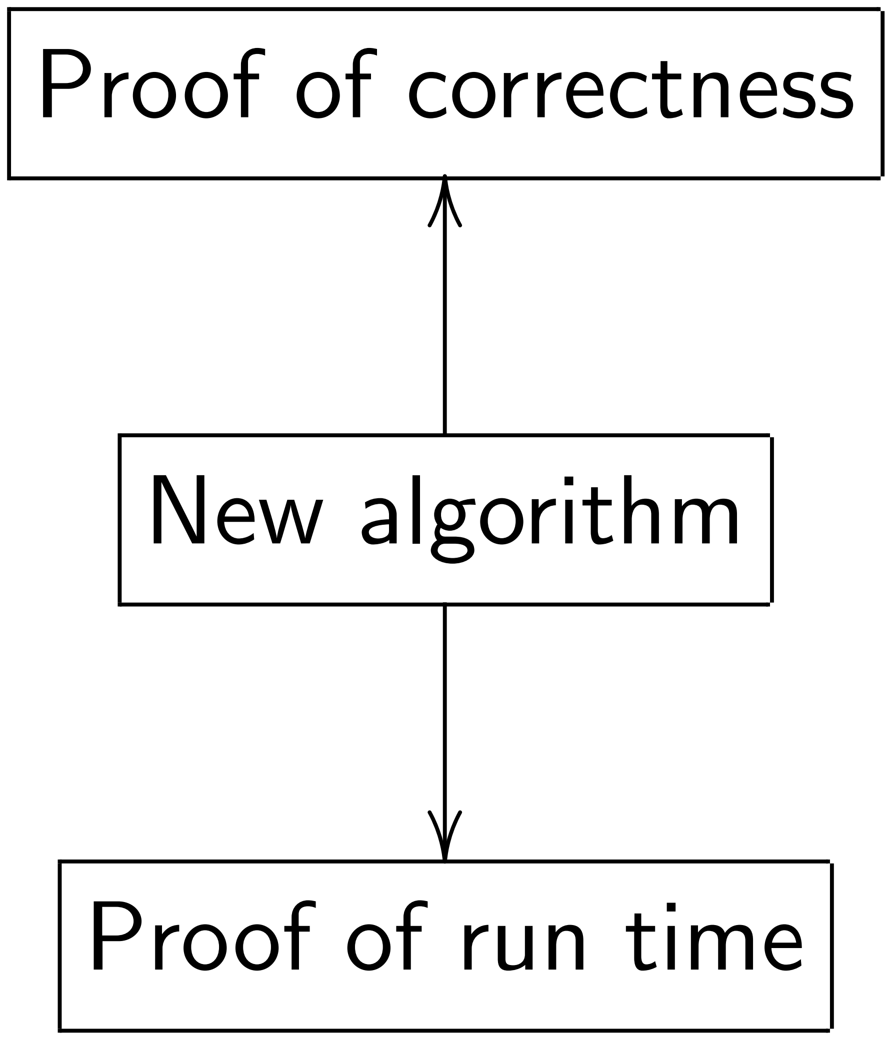
rform

.

ent?

Interlude: Algorithm design

Textbook algorithm analysis:

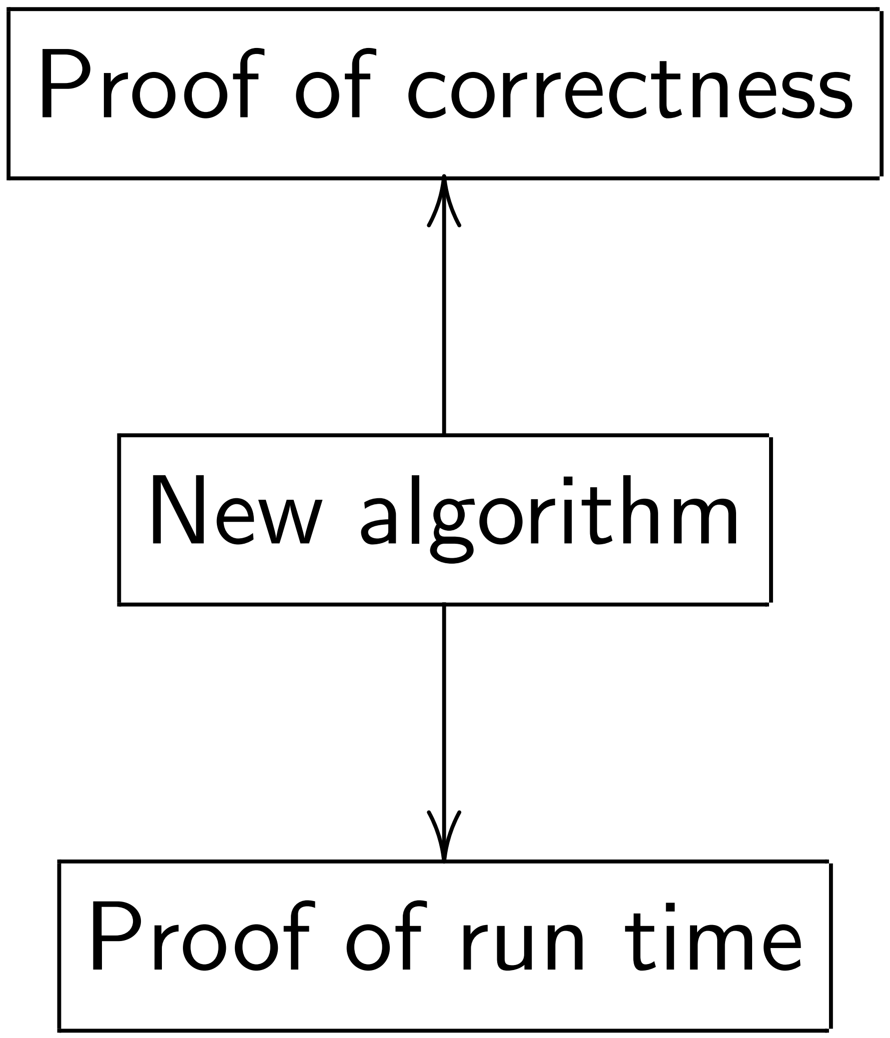


Mislead students into thinking that best algorithm = best *proven* algorithm.

Reality: state-of-the-art cryptanalytic algorithms are almost never proven.

Interlude: Algorithm design

Textbook algorithm analysis:

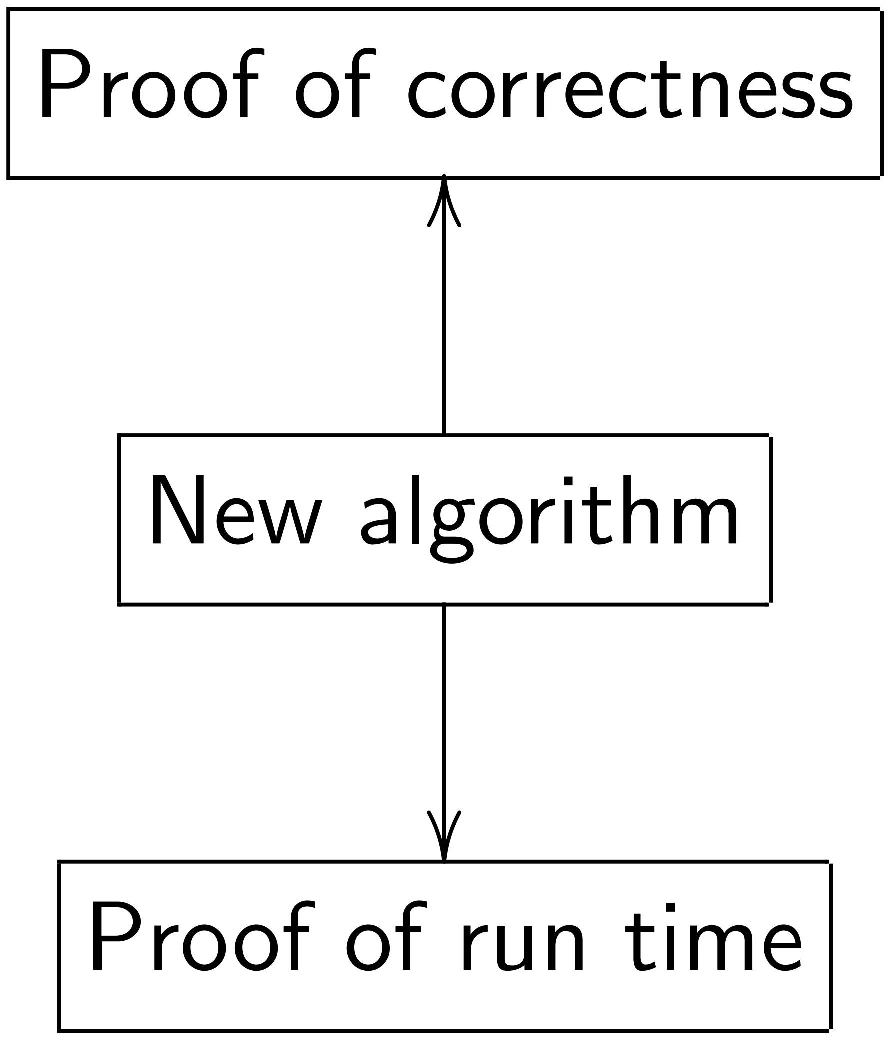


Mislead students into thinking that best algorithm = best *proven* algorithm.

Reality: state-of-the-art cryptanalytic algorithms are almost never proven.

Interlude: Algorithm design

Textbook algorithm analysis:



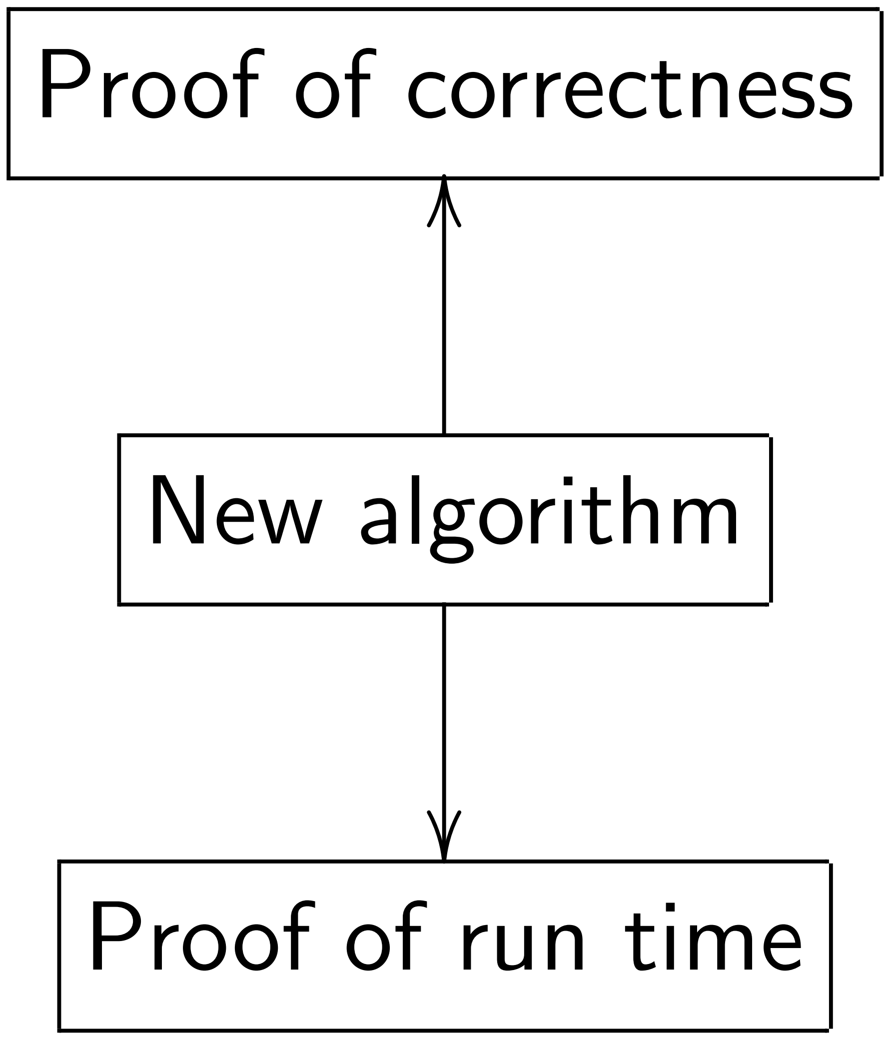
Mislead students into thinking that best algorithm = best *proven* algorithm.

Reality: state-of-the-art cryptanalytic algorithms are almost never proven.

Ignorant response: "Work harder, find proofs!"

Interlude: Algorithm design

Textbook algorithm analysis:



Mislead students into thinking that best algorithm = best *proven* algorithm.

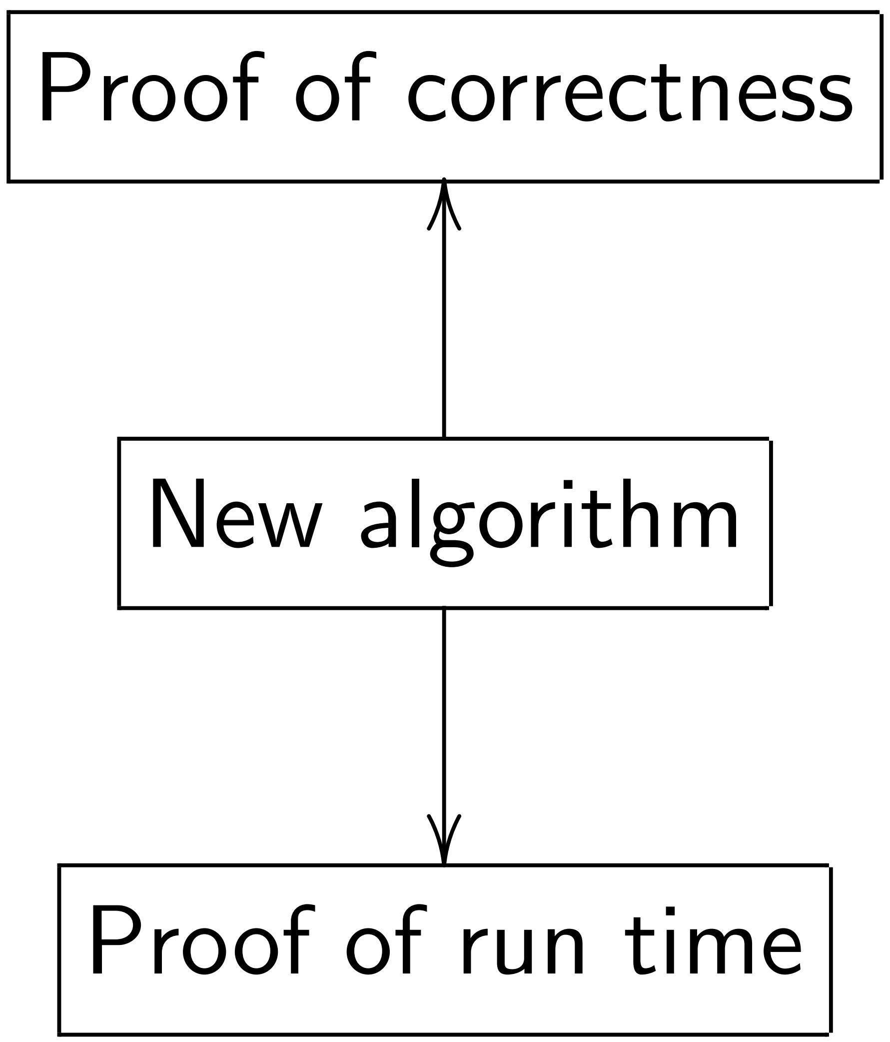
Reality: state-of-the-art cryptanalytic algorithms are almost never proven.

Ignorant response: "Work harder, find proofs!"

Consensus of the experts: proofs probably do not *exist* for most of these algorithms. So demanding proofs is silly.

Interlude: Algorithm design

Textbook algorithm analysis:



Mislead students into thinking that best algorithm = best *proven* algorithm.

Reality: state-of-the-art cryptanalytic algorithms are almost never proven.

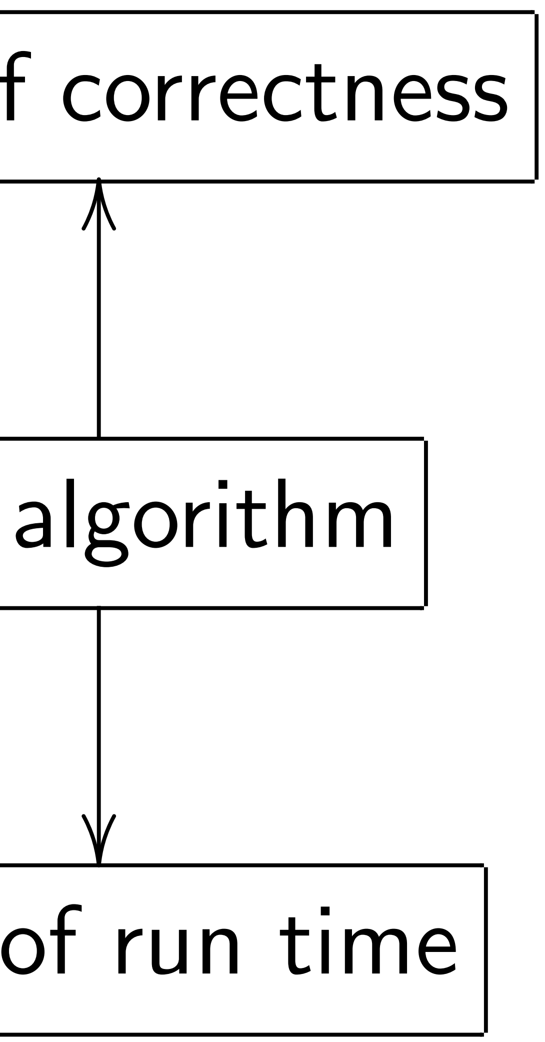
Ignorant response: "Work harder, find proofs!"

Consensus of the experts: proofs probably do not *exist* for most of these algorithms. So demanding proofs is silly.

Without proofs, how do we analyze correctness+speed? Answer: Real algorithm analysis relies critically on heuristics and **computer experiments**.

e: Algorithm design

k algorithm analysis:



students into thinking
t algorithm =
ven algorithm.

Reality: state-of-the-art
cryptanalytic algorithms
are almost never proven.

Ignorant response:
"Work harder, find proofs!"

Consensus of the experts:
proofs probably do not *exist*
for most of these algorithms.
So demanding proofs is silly.

Without proofs, how do we
analyze correctness+speed?
Answer: Real algorithm analysis
relies critically on heuristics and
computer experiments.

What ab

Want to
quantum
to figure
against

hm design

m analysis:

ess

e

nto thinking
m =
thm.

Reality: state-of-the-art cryptanalytic algorithms are almost never proven.

Ignorant response: "Work harder, find proofs!"

Consensus of the experts: proofs probably do not *exist* for most of these algorithms. So demanding proofs is silly.

Without proofs, how do we analyze correctness+speed? Answer: Real algorithm analysis relies critically on heuristics and **computer experiments**.

What about quant

Want to analyze, c
quantum algorithm
to figure out safe c
against *future* qua

:

ng

Reality: state-of-the-art cryptanalytic algorithms are almost never proven.

Ignorant response: "Work harder, find proofs!"

Consensus of the experts: proofs probably do not *exist* for most of these algorithms. So demanding proofs is silly.

Without proofs, how do we analyze correctness+speed? Answer: Real algorithm analysis relies critically on heuristics and **computer experiments**.

What about quantum algori

Want to analyze, optimize quantum algorithms *today* to figure out safe crypto against *future* quantum atta

Reality: state-of-the-art cryptanalytic algorithms are almost never proven.

Ignorant response: "Work harder, find proofs!"

Consensus of the experts: proofs probably do not *exist* for most of these algorithms. So demanding proofs is silly.

Without proofs, how do we analyze correctness+speed? Answer: Real algorithm analysis relies critically on heuristics and **computer experiments**.

What about quantum algorithms?

Want to analyze, optimize quantum algorithms *today* to figure out safe crypto against *future* quantum attack.

Reality: state-of-the-art cryptanalytic algorithms are almost never proven.

Ignorant response: "Work harder, find proofs!"

Consensus of the experts: proofs probably do not *exist* for most of these algorithms. So demanding proofs is silly.

Without proofs, how do we analyze correctness+speed? Answer: Real algorithm analysis relies critically on heuristics and **computer experiments**.

What about quantum algorithms?

Want to analyze, optimize quantum algorithms *today* to figure out safe crypto against *future* quantum attack.

1. Simulate *tiny* q. computer?

$\Rightarrow$ Huge extrapolation errors.

Reality: state-of-the-art cryptanalytic algorithms are almost never proven.

Ignorant response: "Work harder, find proofs!"

Consensus of the experts: proofs probably do not *exist* for most of these algorithms. So demanding proofs is silly.

Without proofs, how do we analyze correctness+speed? Answer: Real algorithm analysis relies critically on heuristics and **computer experiments**.

What about quantum algorithms?

Want to analyze, optimize quantum algorithms *today* to figure out safe crypto against *future* quantum attack.

1. Simulate *tiny* q. computer?
$\Rightarrow$ Huge extrapolation errors.

2. Faster algorithm-specific simulation? Yes, sometimes.

Reality: state-of-the-art cryptanalytic algorithms are almost never proven.

Ignorant response: "Work harder, find proofs!"

Consensus of the experts: proofs probably do not *exist* for most of these algorithms. So demanding proofs is silly.

Without proofs, how do we analyze correctness+speed? Answer: Real algorithm analysis relies critically on heuristics and **computer experiments**.

What about quantum algorithms?

Want to analyze, optimize quantum algorithms *today* to figure out safe crypto against *future* quantum attack.

1. Simulate *tiny* q. computer?
$\Rightarrow$ Huge extrapolation errors.

2. Faster algorithm-specific simulation? Yes, sometimes.

3. Fast **trapdoor simulation.** Simulator (like prover) knows more than the algorithm does.

state-of-the-art
alytic algorithms
ost never proven.

response:
harder, find proofs!"

us of the experts:
robably do not *exist*
of these algorithms.
anding proofs is silly.

proofs, how do we
correctness+speed?
Real algorithm analysis
tically on heuristics and
er experiments.

What about quantum algorithms?

Want to analyze, optimize quantum algorithms *today* to figure out safe crypto against *future* quantum attack.

1. Simulate *tiny* q. computer?
$\Rightarrow$ Huge extrapolation errors.

2. Faster algorithm-specific simulation? Yes, sometimes.

3. Fast **trapdoor simulation.** Simulator (like prover) knows more than the algorithm does.

Quantur

Assume
has n-bi

Generic
finds thi
$\approx 2^n$ eva

1996 Gr
finds thi
$\approx 2^{0.5n}$ c
on super

Cost of
$\approx$ cost c
if cost c

he-art
rithms
roven.

d proofs!"

experts:
o not *exist*
algorithms.
ofs is silly.

ow do we
s+speed?
rithm analysis
heuristics and
ments.

What about quantum algorithms?

Want to analyze, optimize quantum algorithms *today* to figure out safe crypto against *future* quantum attack.

1. Simulate *tiny* q. computer?
$\Rightarrow$ Huge extrapolation errors.

2. Faster algorithm-specific simulation? Yes, sometimes.

3. Fast **trapdoor simulation.** Simulator (like prover) knows more than the algorithm does.

Quantum search (

Assume that funct
has n-bit input, u

Generic brute-forc
finds this root usin
$\approx 2^n$ evaluations o

1996 Grover meth
finds this root usin
$\approx 2^{0.5n}$ quantum e
on superpositions

Cost of quantum e
$\approx$ cost of evaluati
if cost counts qub

r

e.

r

lysis

and

What about quantum algorithms?

Want to analyze, optimize quantum algorithms *today* to figure out safe crypto against *future* quantum attack.

1. Simulate *tiny* q. computer?
$\Rightarrow$ Huge extrapolation errors.

2. Faster algorithm-specific simulation? Yes, sometimes.

3. Fast **trapdoor simulation.** Simulator (like prover) knows more than the algorithm does.

Quantum search (0.5)

Assume that function f has n-bit input, unique root

Generic brute-force search finds this root using $\approx 2^n$ evaluations of f.

1996 Grover method finds this root using $\approx 2^{0.5n}$ quantum evaluations on superpositions of inputs.

Cost of quantum evaluation $\approx$ cost of evaluation of f if cost counts qubit "operati

What about quantum algorithms?

Want to analyze, optimize quantum algorithms *today* to figure out safe crypto against *future* quantum attack.

1. Simulate *tiny* q. computer?
$\Rightarrow$ Huge extrapolation errors.

2. Faster algorithm-specific simulation? Yes, sometimes.

3. Fast **trapdoor simulation.** Simulator (like prover) knows more than the algorithm does.

<u>Quantum search (0.5)</u>

Assume that function f has n-bit input, unique root.

Generic brute-force search finds this root using $\approx 2^n$ evaluations of f.

1996 Grover method finds this root using $\approx 2^{0.5n}$ quantum evaluations of f on superpositions of inputs.

Cost of quantum evaluation of f $\approx$ cost of evaluation of f if cost counts qubit "operations".

bout quantum algorithms?

analyze, optimize
n algorithms *today*
e out safe crypto
future quantum attack.

late *tiny* q. computer?
e extrapolation errors.

er algorithm-specific
on? Yes, sometimes.

trapdoor simulation.
or (like prover) knows
an the algorithm does.

Quantum search (0.5)

Assume that function f
has n-bit input, unique root.

Generic brute-force search
finds this root using
$\approx 2^n$ evaluations of f.

1996 Grover method
finds this root using
$\approx 2^{0.5n}$ quantum evaluations of f
on superpositions of inputs.

Cost of quantum evaluation of f
$\approx$ cost of evaluation of f
if cost counts qubit "operations".

Easily ac
different
and # n
Faster if
but typic
Most int

tum algorithms?

optimize
ns *today*
crypto
ntum attack.

q. computer?
tion errors.

m-specific
sometimes.

simulation.
over) knows
orithm does.

Quantum search (0.5)

Assume that function f
has n-bit input, unique root.

Generic brute-force search
finds this root using
$\approx 2^n$ evaluations of f.

1996 Grover method
finds this root using
$\approx 2^{0.5n}$ quantum evaluations of f
on superpositions of inputs.

Cost of quantum evaluation of f
$\approx$ cost of evaluation of f
if cost counts qubit "operations".

Easily adapt to ha
different # of roo
and # not known
Faster if # is larg
but typically # is
Most interesting:

thms?

ack.

er?

s.

.

n.

ys

es.

Quantum search (0.5)

Assume that function f
has n-bit input, unique root.

Generic brute-force search
finds this root using
$\approx 2^n$ evaluations of f.

1996 Grover method
finds this root using
$\approx 2^{0.5n}$ quantum evaluations of f
on superpositions of inputs.

Cost of quantum evaluation of f
$\approx$ cost of evaluation of f
if cost counts qubit “operations”.

Easily adapt to handle
different # of roots,
and # not known in advanc
Faster if # is large,
but typically # is not very la
Most interesting: $\# \in \{0, 1$

Quantum search (0.5)

Assume that function f has n-bit input, unique root.

Generic brute-force search finds this root using $\approx 2^n$ evaluations of f.

1996 Grover method finds this root using $\approx 2^{0.5n}$ quantum evaluations of f on superpositions of inputs.

Cost of quantum evaluation of f $\approx$ cost of evaluation of f if cost counts qubit “operations”.

Easily adapt to handle different # of roots, and # not known in advance. Faster if # is large, but typically # is not very large. Most interesting: $\# \in \{0, 1\}$.

Quantum search (0.5)

Assume that function f has n-bit input, unique root.

Generic brute-force search finds this root using $\approx 2^n$ evaluations of f.

1996 Grover method finds this root using $\approx 2^{0.5n}$ quantum evaluations of f on superpositions of inputs.

Cost of quantum evaluation of f $\approx$ cost of evaluation of f if cost counts qubit "operations".

Easily adapt to handle different # of roots, and # not known in advance. Faster if # is large, but typically # is not very large. Most interesting: $\# \in \{0, 1\}$.

Apply to the function $J \mapsto \Sigma(J) - t$ where $\Sigma(J) = \sum_{i \in J} x_i$.

Cost $2^{0.5n}$ to find root (i.e., to find indices of subsequence of $x_1, \ldots, x_n$ with sum t) or to decide that no root exists. We suppress poly factors in cost.

m search (0.5)

that function f
t input, unique root.

brute-force search
s root using
luations of f.

over method
s root using
quantum evaluations of f
rpositions of inputs.

quantum evaluation of f
of evaluation of f
ounts qubit "operations".

Easily adapt to handle
different # of roots,
and # not known in advance.
Faster if # is large,
but typically # is not very large.
Most interesting: $\# \in \{0, 1\}$.

Apply to the function
$J \mapsto \Sigma(J) - t$ where
$\Sigma(J) = \sum_{i \in J} x_i$.

Cost $2^{0.5n}$ to find root (i.e.,
to find indices of subsequence
of $x_1, \ldots, x_n$ with sum t)
or to decide that no root exists.
We suppress poly factors in cost.

Algorith

Represen
integer b

n bits ar
to store

n qubits
a superp
2^n comp
$a_0, \ldots, a$
$|a_0|^2 + \cdot$
Measurin
has chan

Start fro
i.e., $a_J =$

0.5)

tion f
nique root.

e search
ng
of f.

od
ng
evaluations of f
of inputs.

evaluation of f
on of f
it "operations".

Easily adapt to handle different # of roots, and # not known in advance. Faster if # is large, but typically # is not very large. Most interesting: $\# \in \{0,1\}$.

Apply to the function $J \mapsto \Sigma(J) - t$ where $\Sigma(J) = \sum_{i \in J} x_i$.

Cost $2^{0.5n}$ to find root (i.e., to find indices of subsequence of $x_1, \ldots, x_n$ with sum t) or to decide that no root exists. We suppress poly factors in cost.

Algorithm details f

Represent $J \subseteq \{1,$
integer between 0

n bits are enough
to store one such

n qubits store mu
a superposition ov
2^n complex amplit
$a_0, \ldots, a_{2^n-1}$ with
$|a_0|^2 + \cdots + |a_{2^n}$
Measuring these n
has chance $|a_J|^2$ t

Start from uniform
i.e., $a_J = 1/2^{n/2}$ f

.

s of f

of f

ions".

Easily adapt to handle
different # of roots,
and # not known in advance.
Faster if # is large,
but typically # is not very large.
Most interesting: $\# \in \{0, 1\}$.

Apply to the function
$J \mapsto \Sigma(J) - t$ where
$\Sigma(J) = \sum_{i \in J} x_i$.

Cost $2^{0.5n}$ to find root (i.e.,
to find indices of subsequence
of $x_1, \ldots, x_n$ with sum t)
or to decide that no root exists.
We suppress poly factors in cost.

Algorithm details for unique

Represent $J \subseteq \{1, \ldots, n\}$ as
integer between 0 and $2^n -$

n bits are enough space
to store one such integer.

n qubits store much more,
a superposition over sets J:
2^n complex amplitudes
$a_0, \ldots, a_{2^n-1}$ with
$|a_0|^2 + \cdots + |a_{2^n-1}|^2 = 1$.
Measuring these n qubits
has chance $|a_J|^2$ to produce

Start from uniform superpos
i.e., $a_J = 1/2^{n/2}$ for all J.

Easily adapt to handle
different # of roots,
and # not known in advance.
Faster if # is large,
but typically # is not very large.
Most interesting: $\# \in \{0, 1\}$.

Apply to the function
$J \mapsto \Sigma(J) - t$ where
$\Sigma(J) = \sum_{i \in J} x_i$.

Cost $2^{0.5n}$ to find root (i.e.,
to find indices of subsequence
of $x_1, \ldots, x_n$ with sum t)
or to decide that no root exists.
We suppress poly factors in cost.

Algorithm details for unique root:

Represent $J \subseteq \{1, \ldots, n\}$ as an
integer between 0 and $2^n - 1$.

n bits are enough space
to store one such integer.

n qubits store much more,
a superposition over sets J:
2^n complex amplitudes
$a_0, \ldots, a_{2^n-1}$ with
$|a_0|^2 + \cdots + |a_{2^n-1}|^2 = 1$.
Measuring these n qubits
has chance $|a_J|^2$ to produce J.

Start from uniform superposition,
i.e., $a_J = 1/2^{n/2}$ for all J.

dapt to handle
# of roots,
not known in advance.
is large,
cally # is not very large.
teresting: $\# \in \{0, 1\}$.

the function
$J) - t$ where
$\sum_{i \in J} x_i$.

^{5n} to find root (i.e.,
ndices of subsequence
$\ldots, x_n$ with sum t)
cide that no root exists.
press poly factors in cost.

Algorithm details for unique root:

Represent $J \subseteq \{1, \ldots, n\}$ as an integer between 0 and $2^n - 1$.

n bits are enough space to store one such integer.

n qubits store much more, a superposition over sets J: 2^n complex amplitudes $a_0, \ldots, a_{2^n-1}$ with $|a_0|^2 + \cdots + |a_{2^n-1}|^2 = 1$. Measuring these n qubits has chance $|a_J|^2$ to produce J.

Start from uniform superposition, i.e., $a_J = 1/2^{n/2}$ for all J.

Step 1:
$b_J = -a$
$b_J = a_J$
This is a
as comp

Step 2:
Set $a \leftarrow$
$b_J = -a$
This is a

Repeat s
about 0.

Measure
With hig
the uniq

ndle
ts,
in advance.
e,
not very large.
$\# \in \{0, 1\}$.

tion
ere

root (i.e.,
subsequence
sum t)
no root exists.
factors in cost.

Algorithm details for unique root:

Represent $J \subseteq \{1, \ldots, n\}$ as an integer between 0 and $2^n - 1$.

n bits are enough space to store one such integer.

n qubits store much more, a superposition over sets J: 2^n complex amplitudes $a_0, \ldots, a_{2^n-1}$ with $|a_0|^2 + \cdots + |a_{2^n-1}|^2 = 1$. Measuring these n qubits has chance $|a_J|^2$ to produce J.

Start from uniform superposition, i.e., $a_J = 1/2^{n/2}$ for all J.

Step 1: Set $a \leftarrow$
$b_J = -a_J$ if $\Sigma(J)$
$b_J = a_J$ otherwise
This is about as ea
as computing Σ.

Step 2: "Grover d
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^{n}$
This is also easy.

Repeat steps 1 an
about $0.58 \cdot 2^{0.5n}$

Measure the n qu
With high probabi
the unique J such

e.

arge.

}.

ce

ists.

cost.

Algorithm details for unique root:

Represent $J \subseteq \{1, \ldots, n\}$ as an integer between 0 and $2^n - 1$.

n bits are enough space to store one such integer.

n qubits store much more, a superposition over sets J: 2^n complex amplitudes $a_0, \ldots, a_{2^n-1}$ with $|a_0|^2 + \cdots + |a_{2^n-1}|^2 = 1$. Measuring these n qubits has chance $|a_J|^2$ to produce J.

Start from uniform superposition, i.e., $a_J = 1/2^{n/2}$ for all J.

Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2 about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this fi
the unique J such that $\Sigma(J$

Algorithm details for unique root:

Represent $J \subseteq \{1, \ldots, n\}$ as an integer between 0 and $2^n - 1$.

n bits are enough space to store one such integer.

n qubits store much more, a superposition over sets J: 2^n complex amplitudes $a_0, \ldots, a_{2^n-1}$ with $|a_0|^2 + \cdots + |a_{2^n-1}|^2 = 1$. Measuring these n qubits has chance $|a_J|^2$ to produce J.

Start from uniform superposition, i.e., $a_J = 1/2^{n/2}$ for all J.

Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2 about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits. With high probability this finds the unique J such that $\Sigma(J) = t$.

m details for unique root:

nt $J \subseteq \{1, \ldots, n\}$ as an
between 0 and $2^n - 1$.

re enough space
one such integer.

s store much more,
position over sets J:
plex amplitudes
a_{2^n-1} with
$\cdots + |a_{2^n-1}|^2 = 1$.
ng these n qubits
nce $|a_J|^2$ to produce J.

om uniform superposition,
$= 1/2^{n/2}$ for all J.

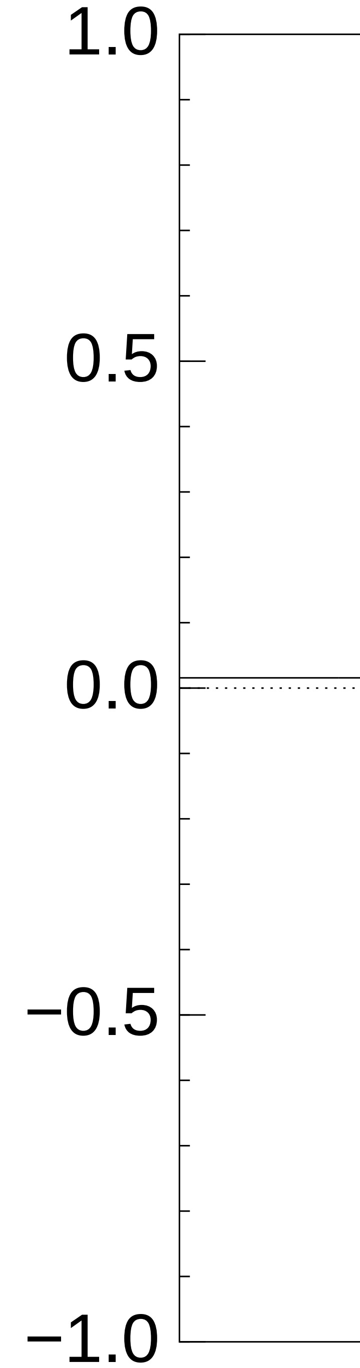
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of
for 3663
after 0 s



for unique root:

$\ldots, n\}$ as an

and $2^n - 1$.

space

integer.

ch more,

er sets J:

tudes

n

$_{-1}|^2 = 1$.

qubits

to produce J.

n superposition,

for all J.

Step 1: Set $a \leftarrow b$ where

$b_J = -a_J$ if $\Sigma(J) = t$,

$b_J = a_J$ otherwise.

This is about as easy

as computing Σ.

Step 2: "Grover diffusion".

Set $a \leftarrow b$ where

$b_J = -a_J + (2/2^n) \sum_I a_I$.

This is also easy.

Repeat steps 1 and 2

about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.

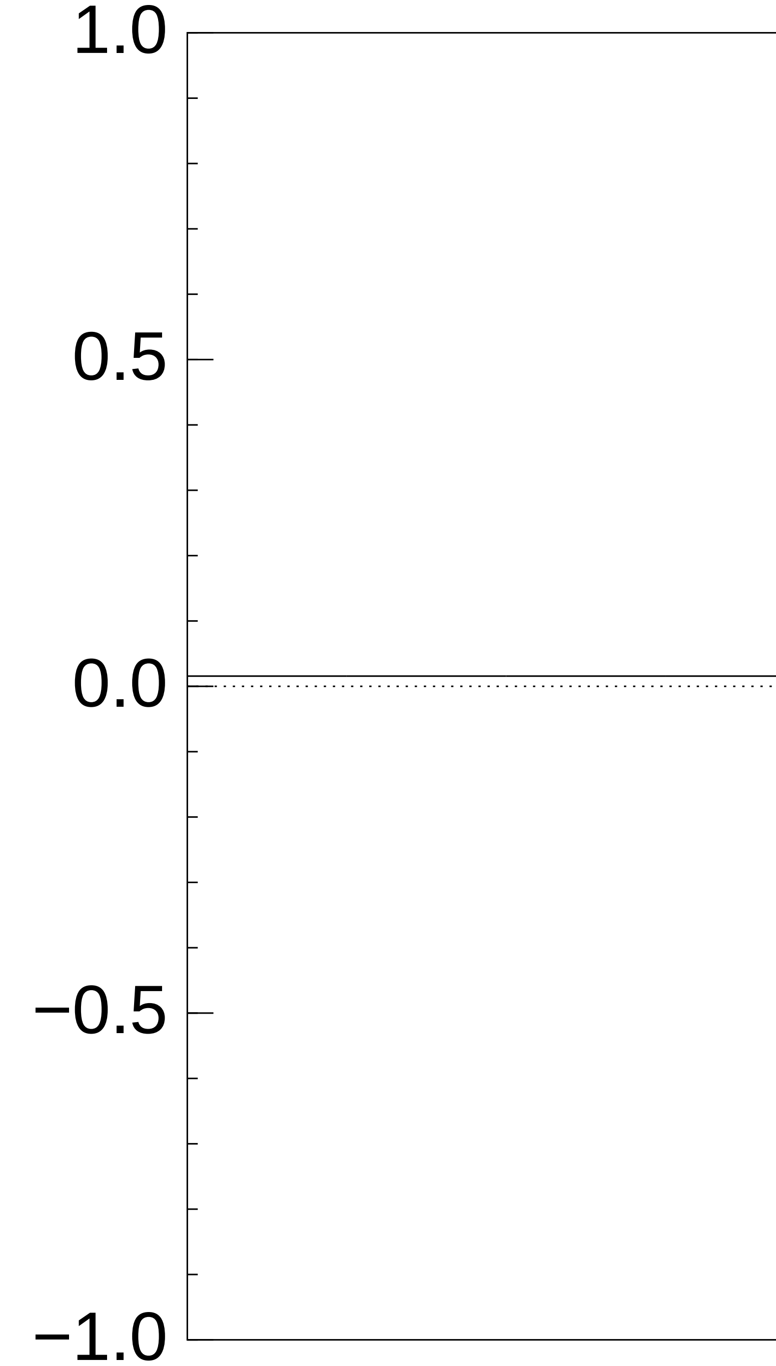
With high probability this finds

the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$

for 36634 example

after 0 steps:



root:

s an

1.

J.

sition,

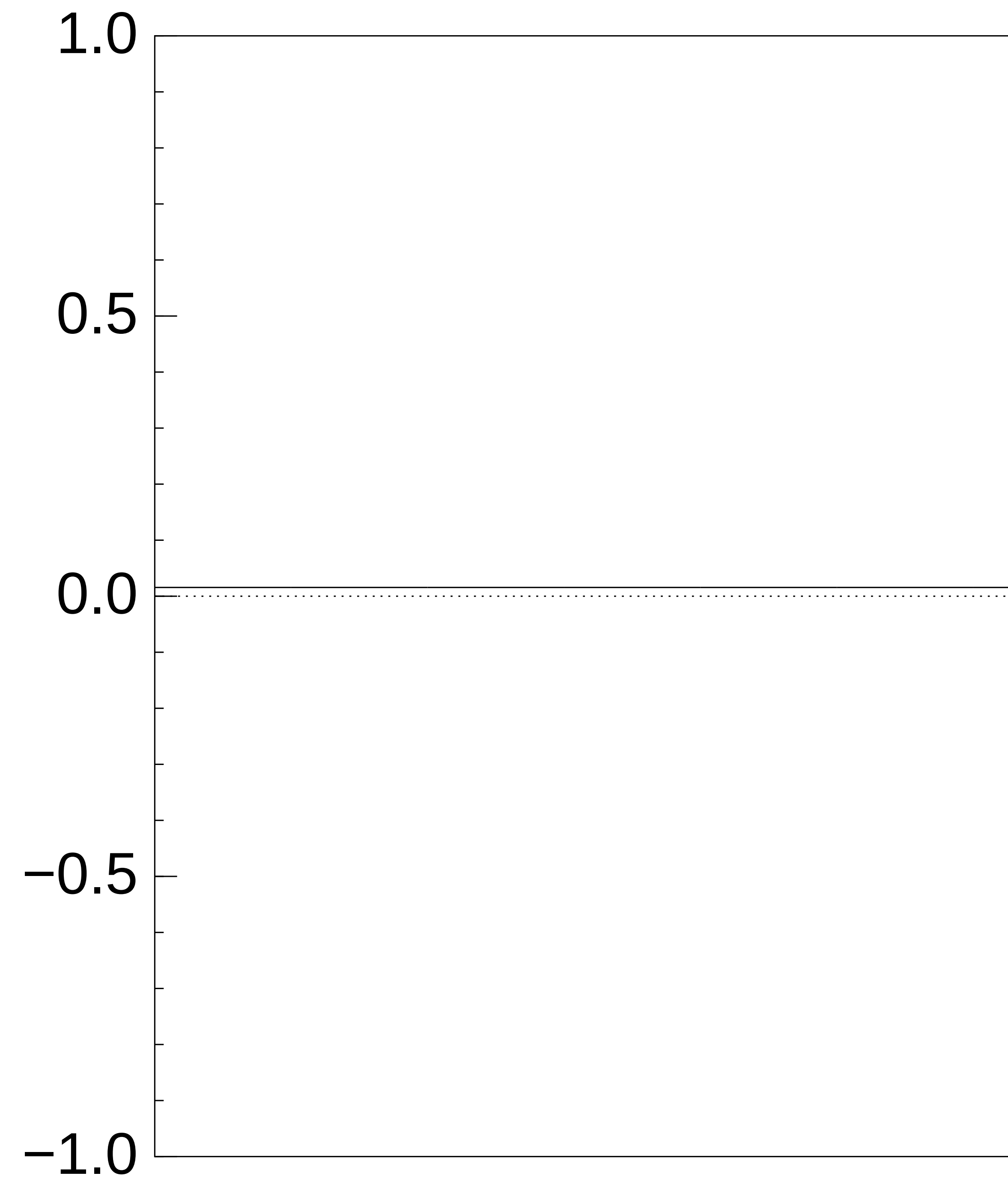
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n =$
after 0 steps:



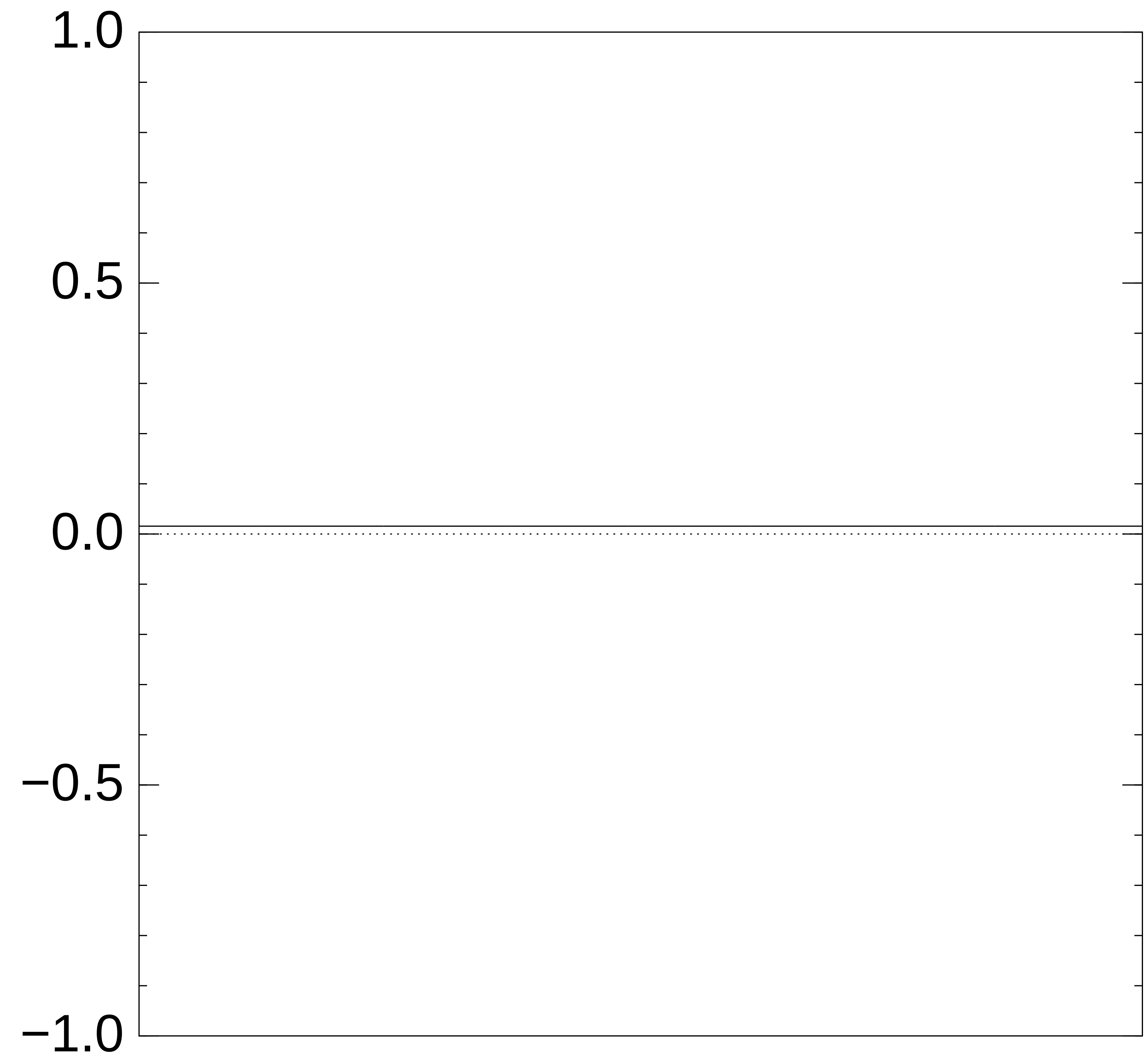
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after 0 steps:



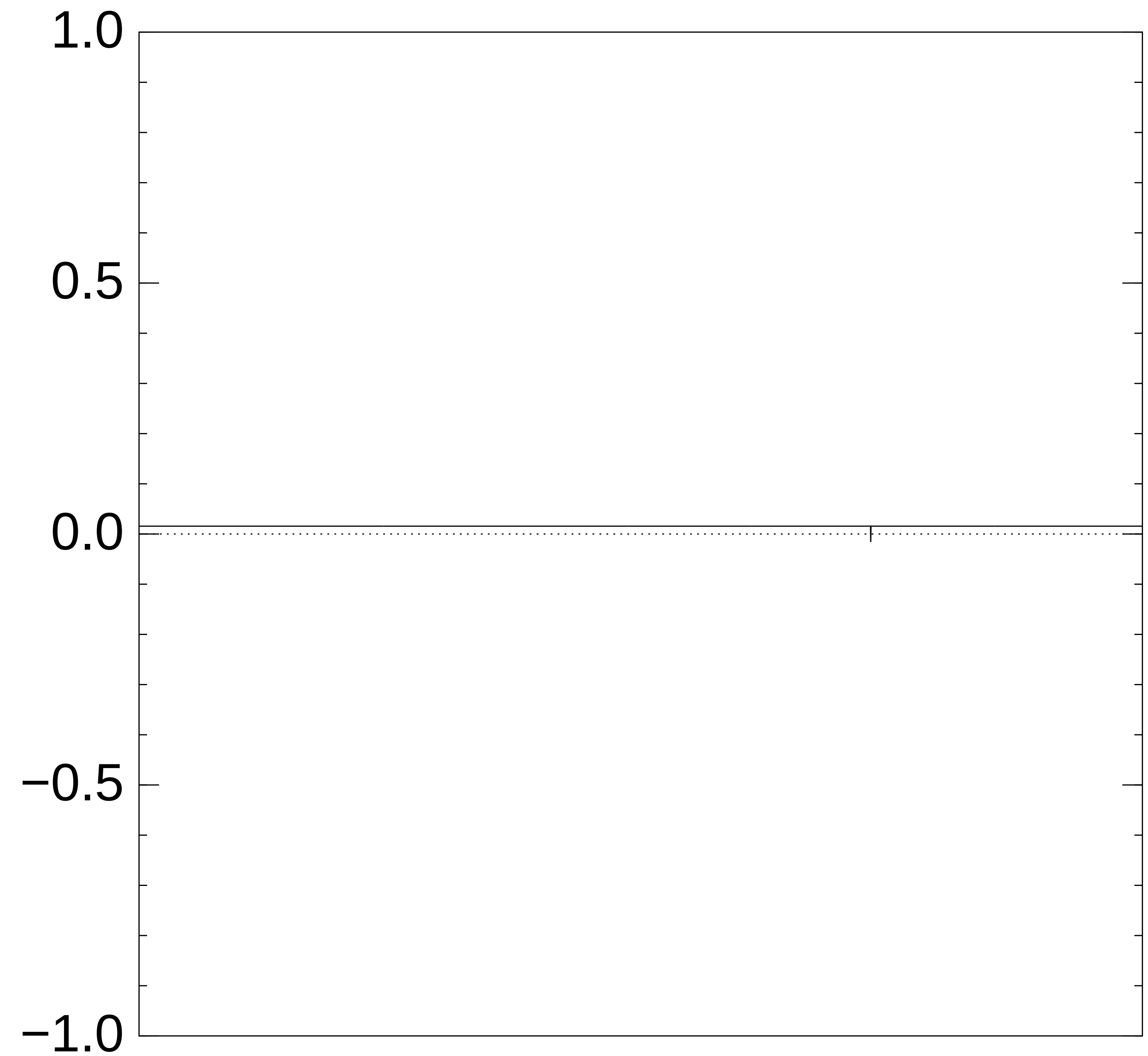
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after Step 1:



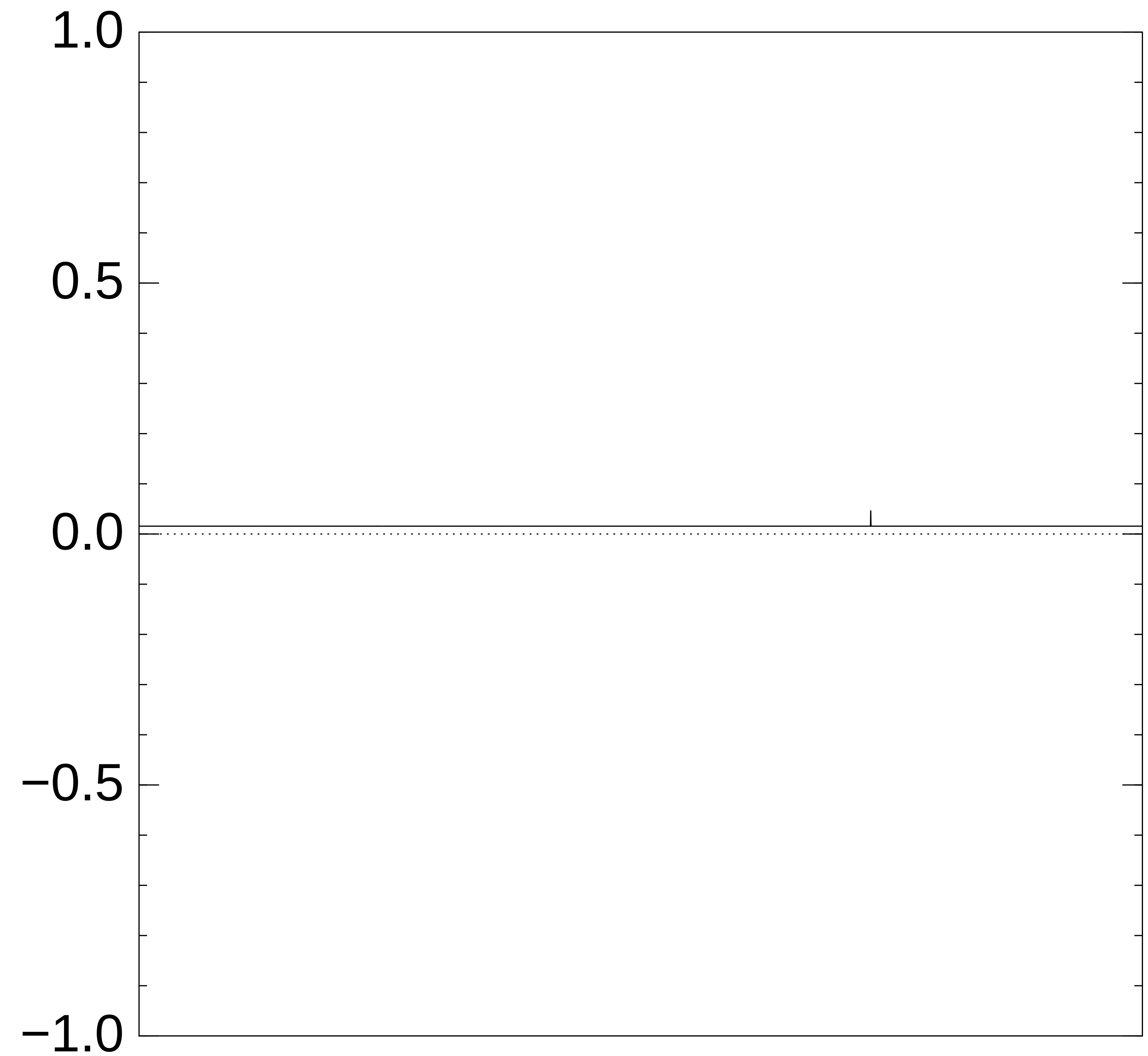
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after Step 1 + Step 2:



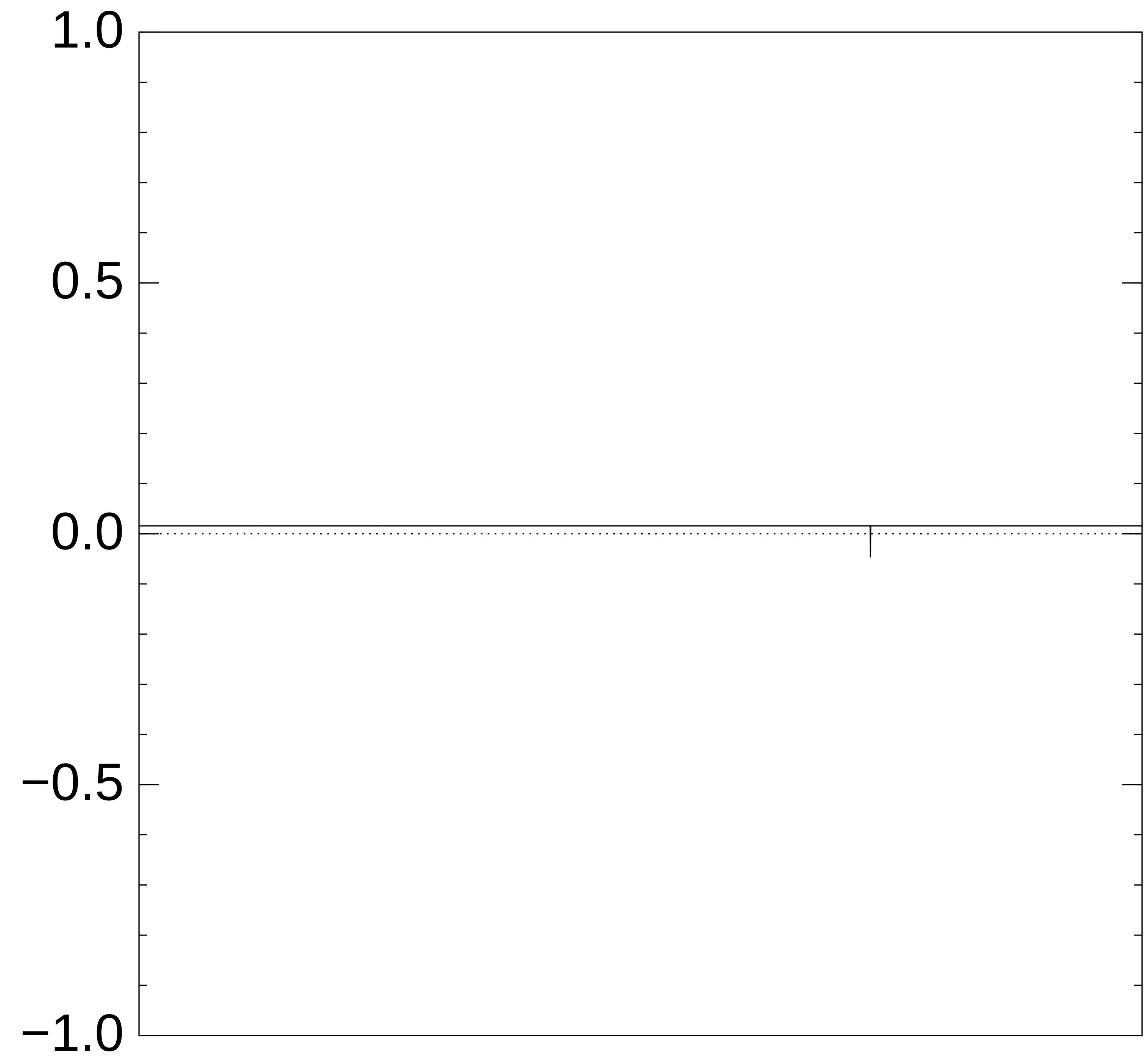
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after Step 1 + Step 2 + Step 1:



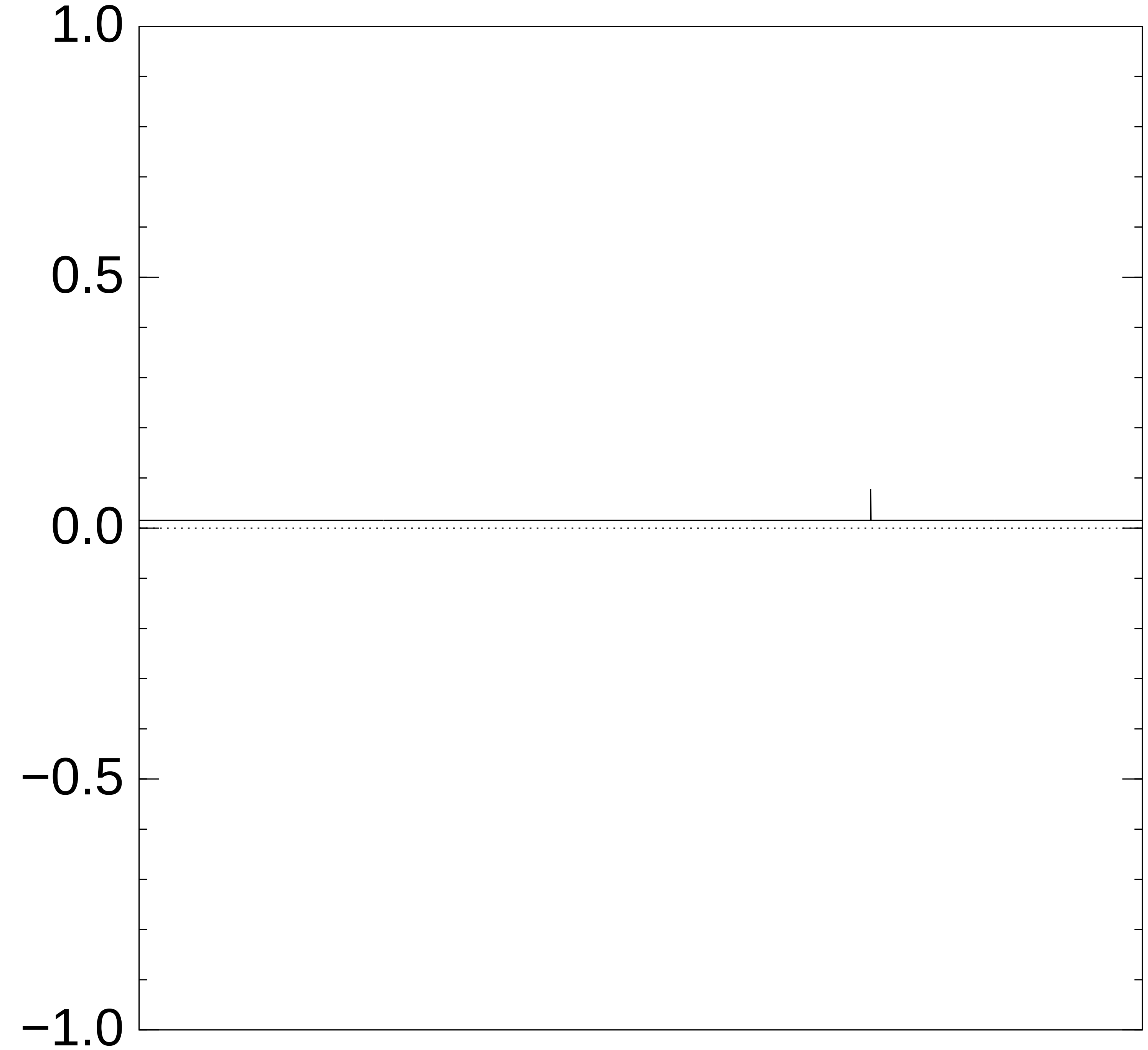
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2 about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$ for 36634 example with $n = 12$ after $2 \times$ (Step 1 + Step 2):



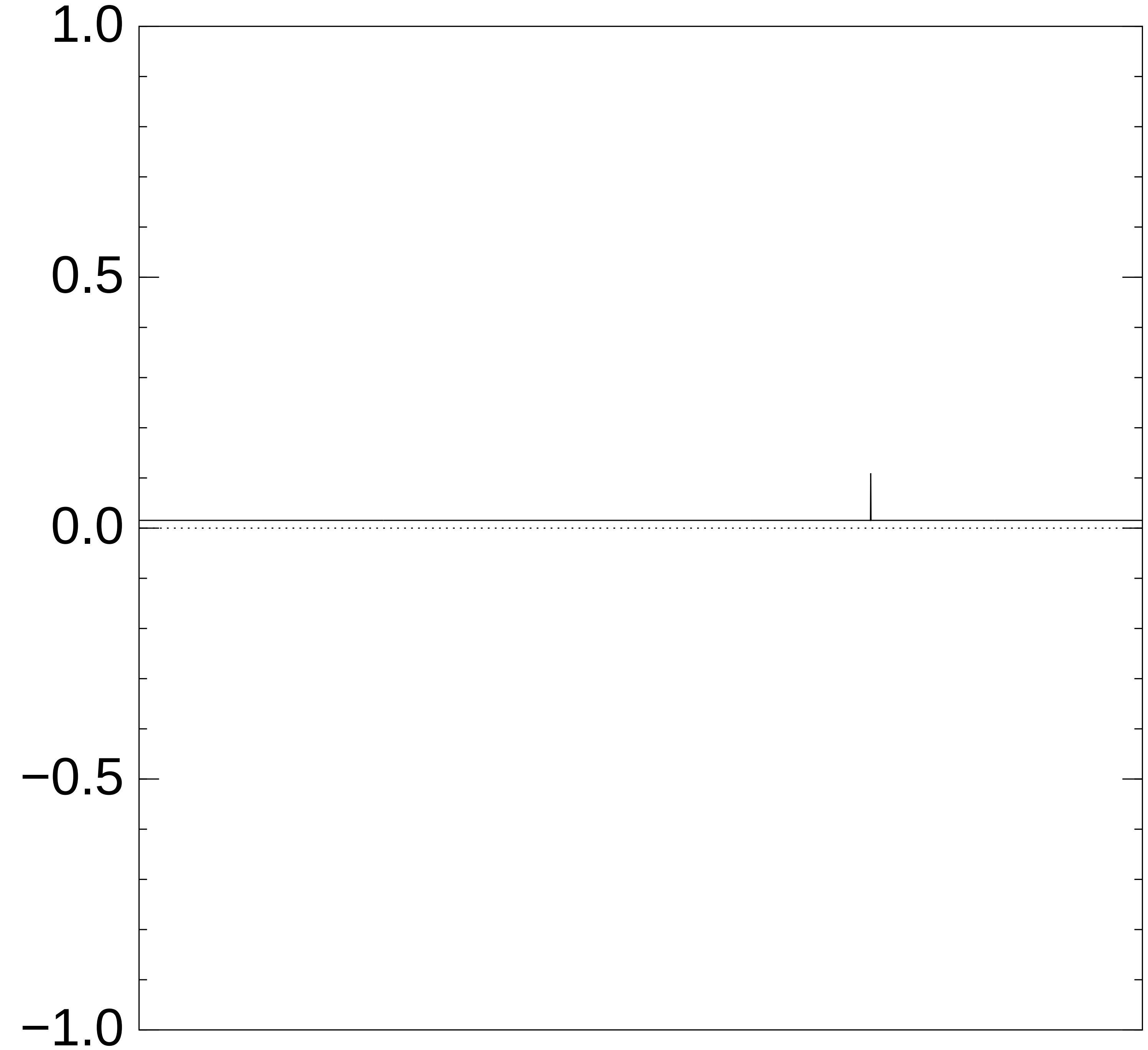
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $3 \times$ (Step 1 + Step 2):



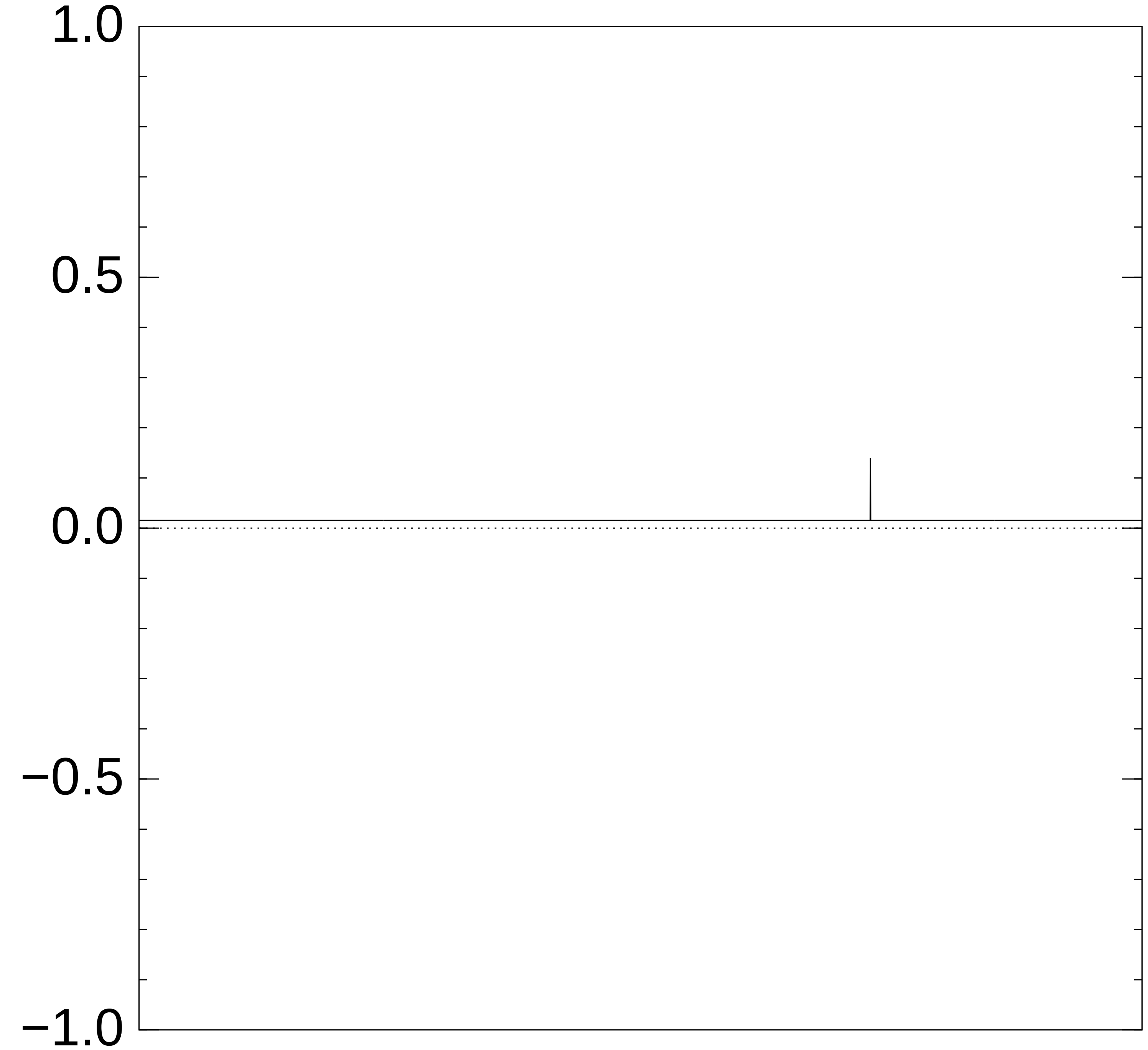
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $4 \times$ (Step 1 + Step 2):



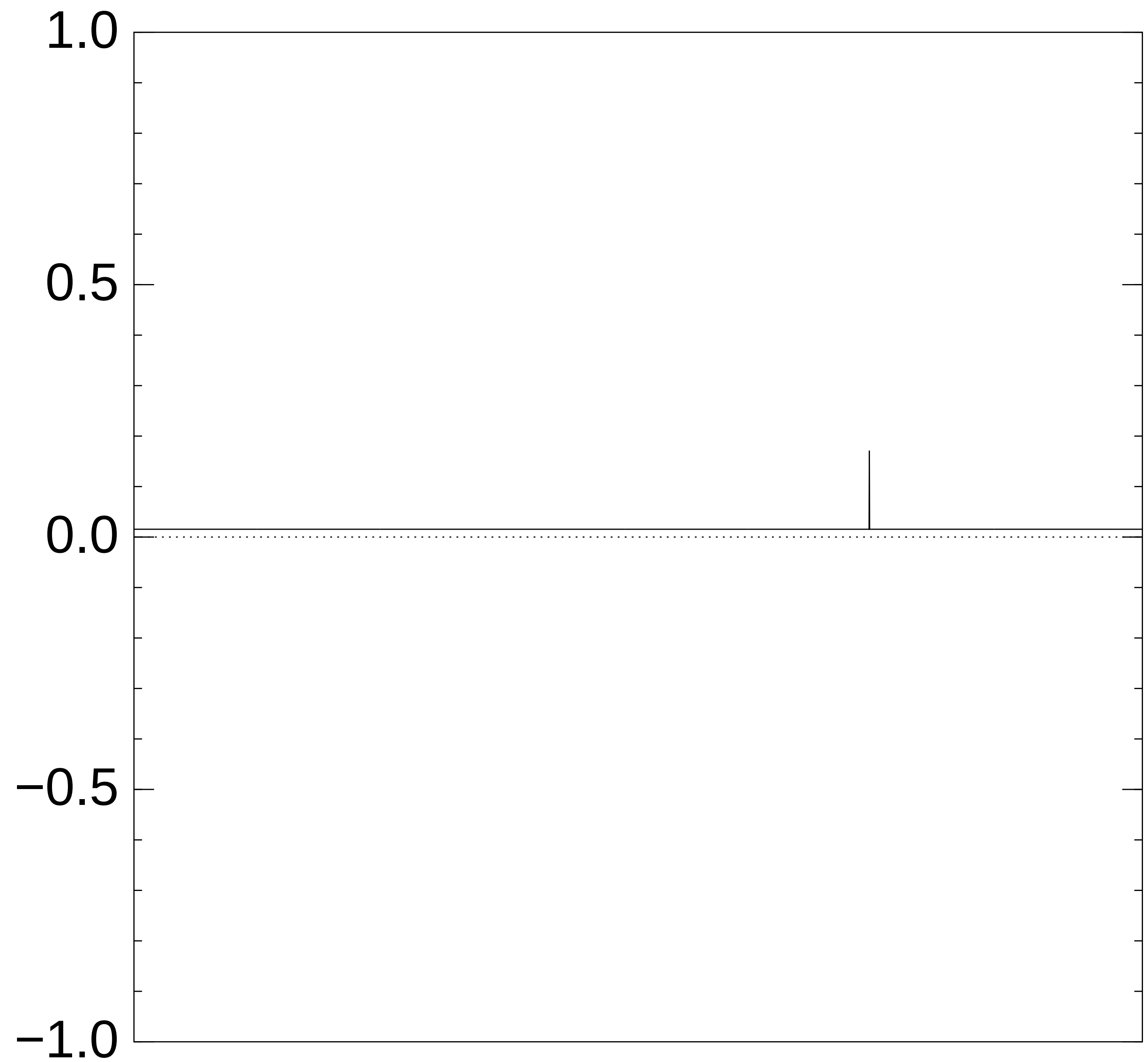
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $5 \times$ (Step 1 + Step 2):



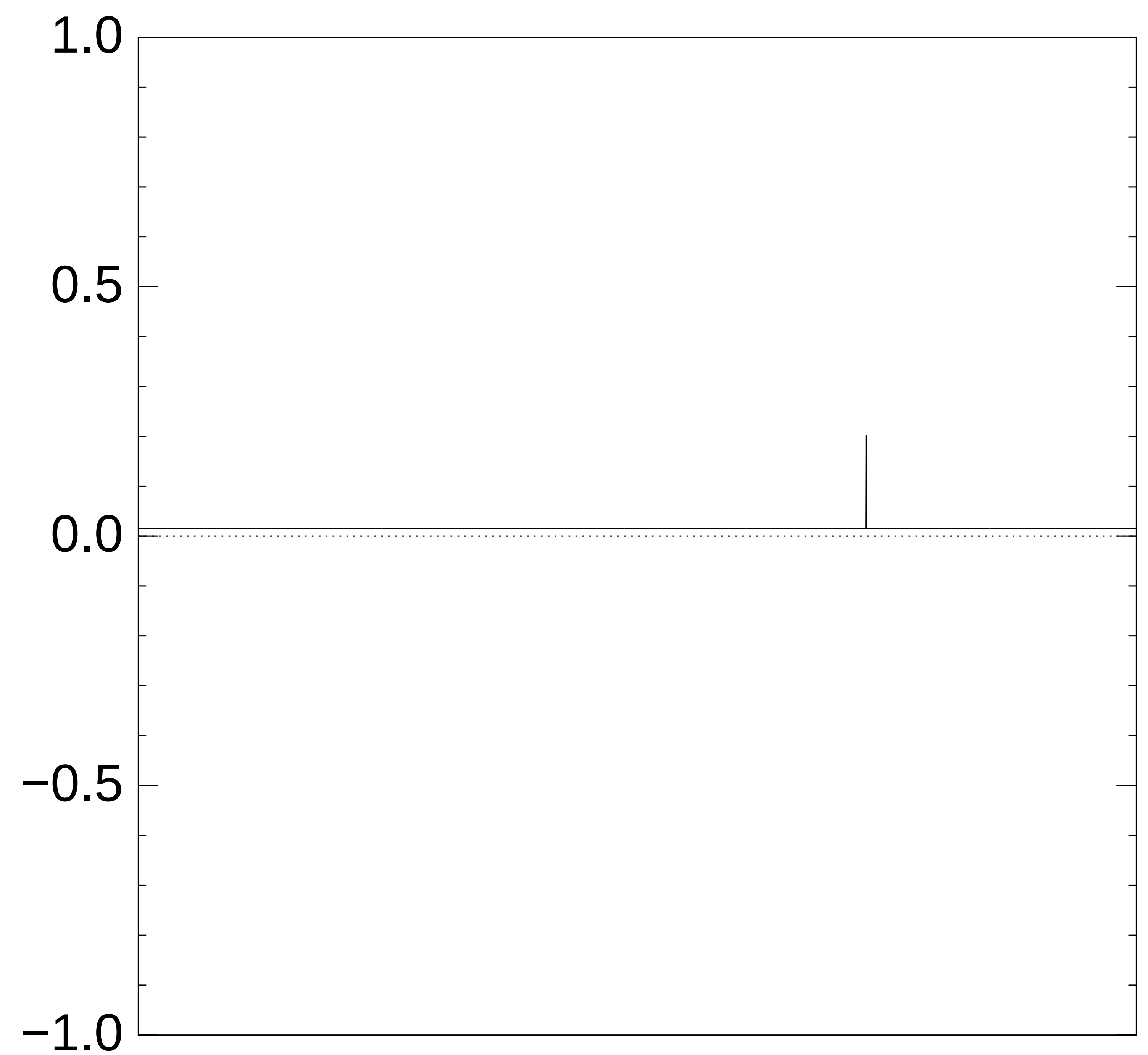
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $6 \times$ (Step $1 +$ Step 2):



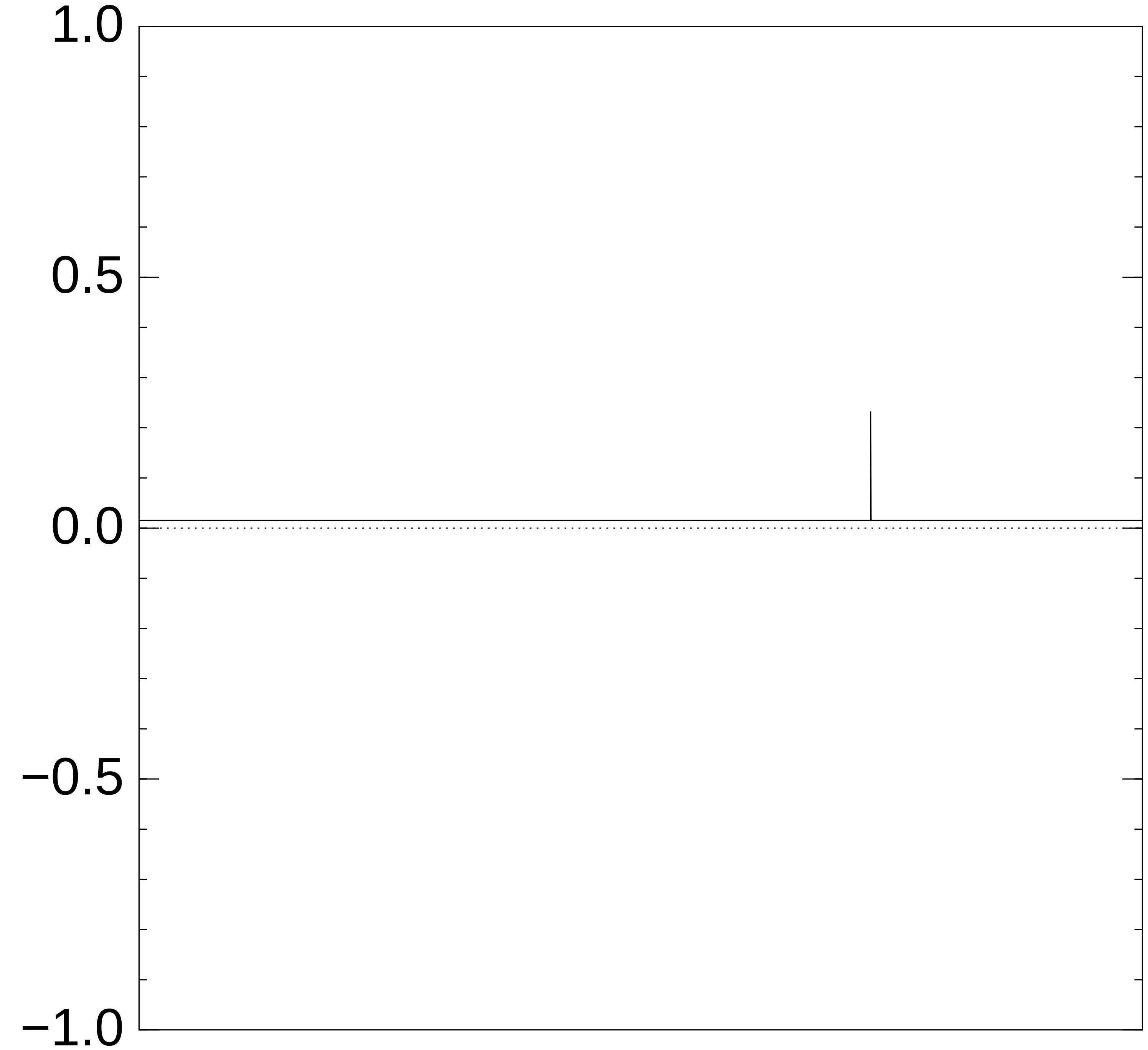
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $7 \times$ (Step 1 + Step 2):



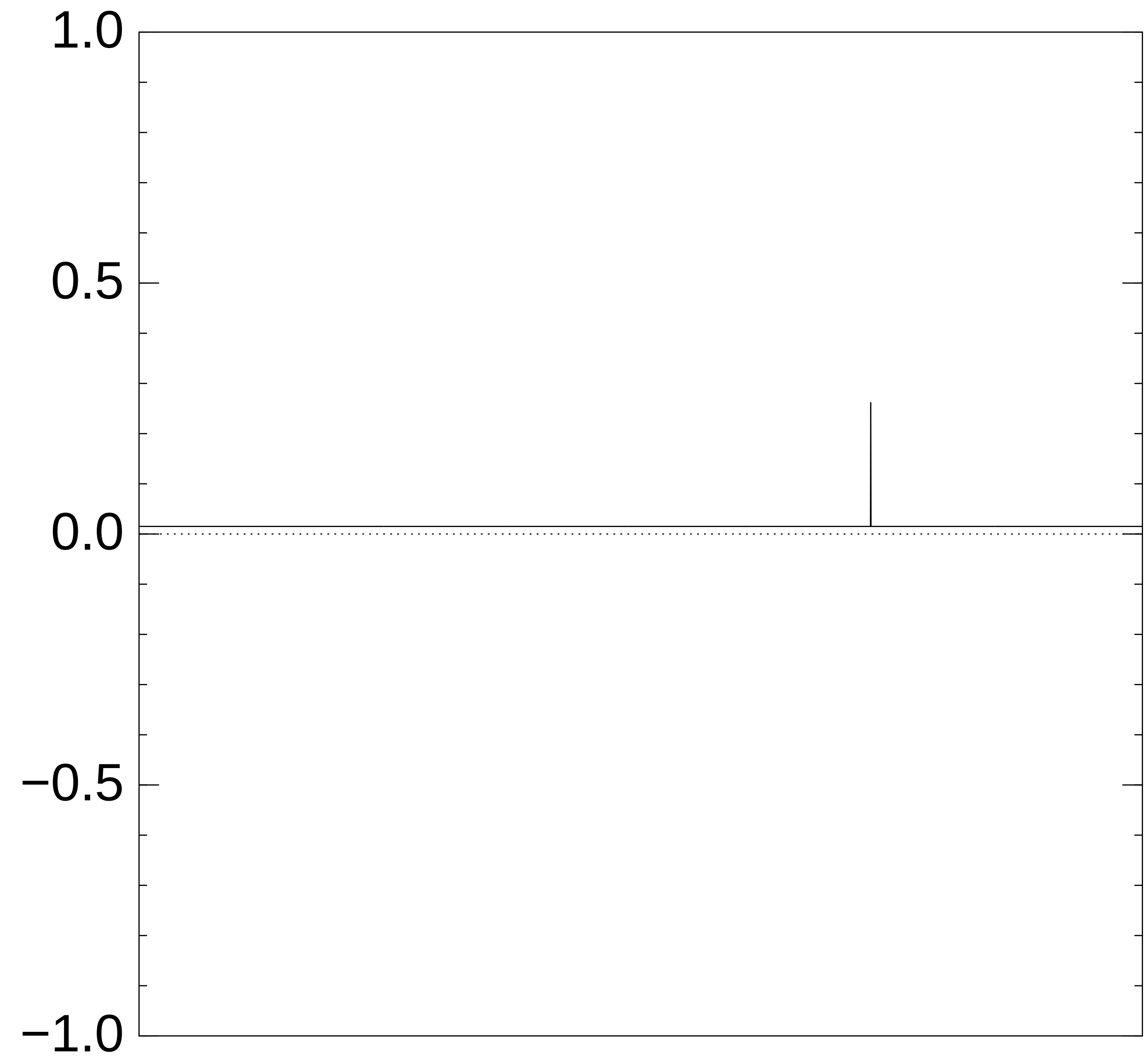
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n)\sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $8 \times$ (Step $1 +$ Step 2):



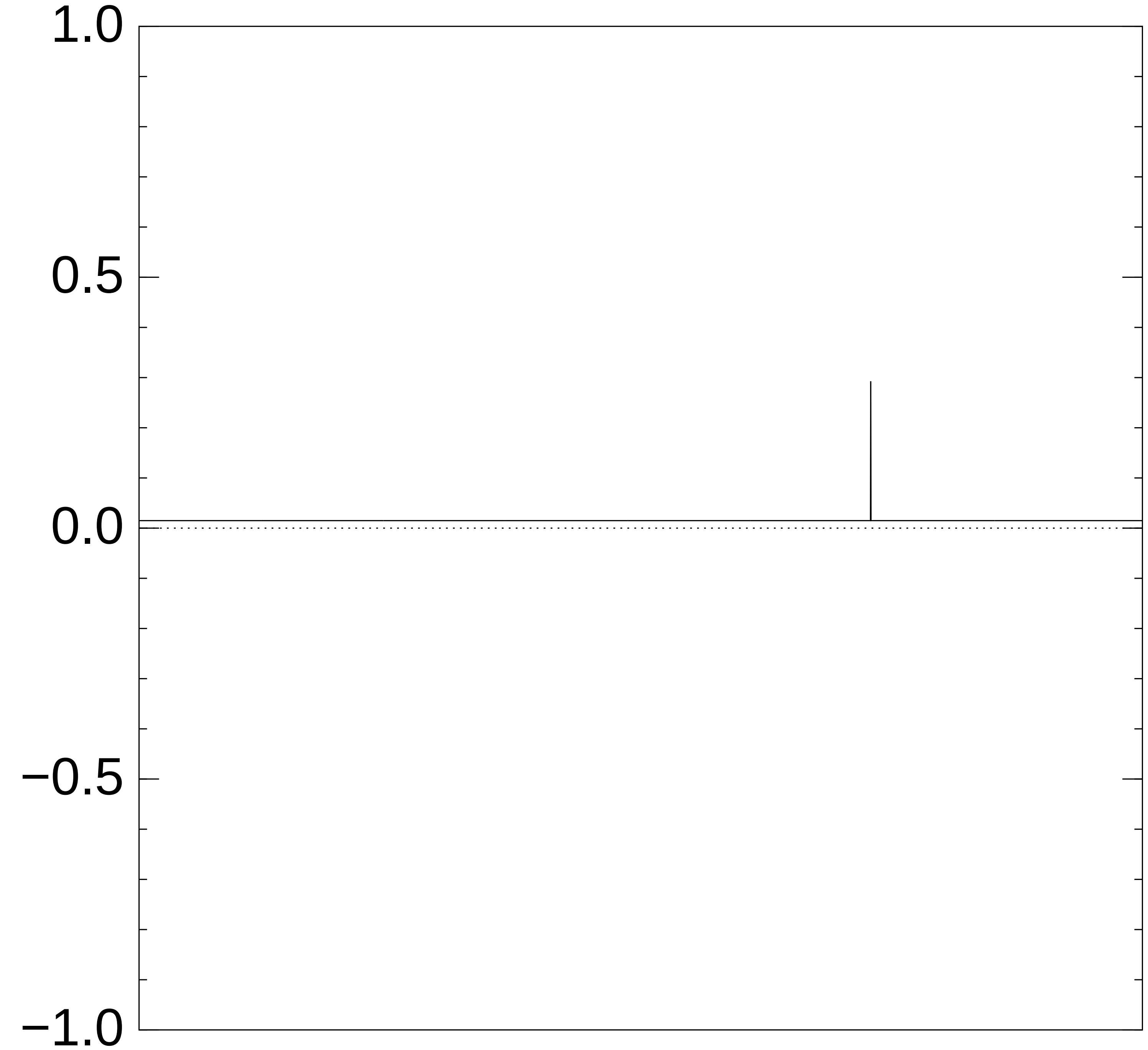
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $9 \times$ (Step 1 + Step 2):



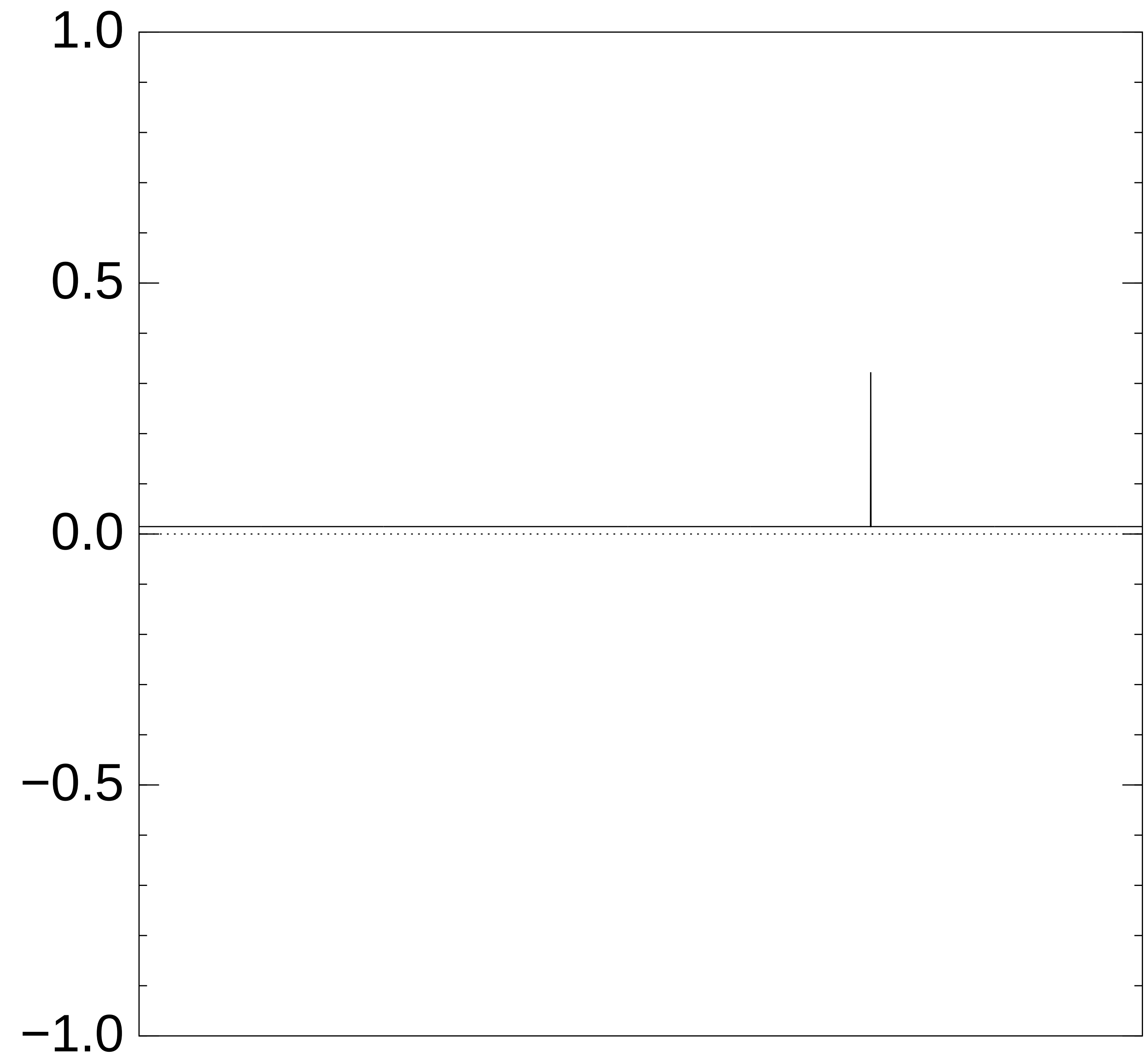
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $10 \times$ (Step 1 + Step 2):



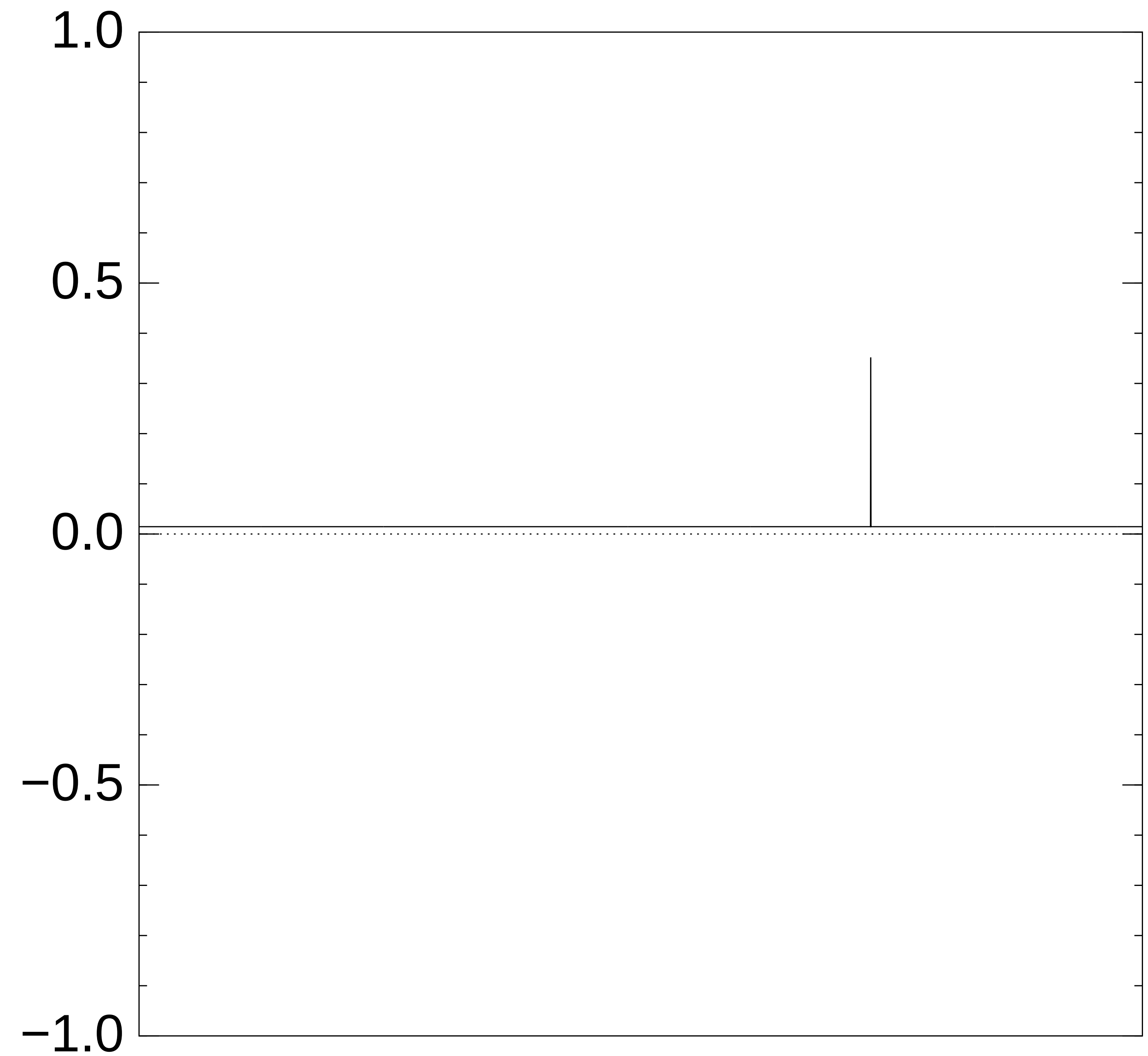
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $11 \times$ (Step 1 + Step 2):



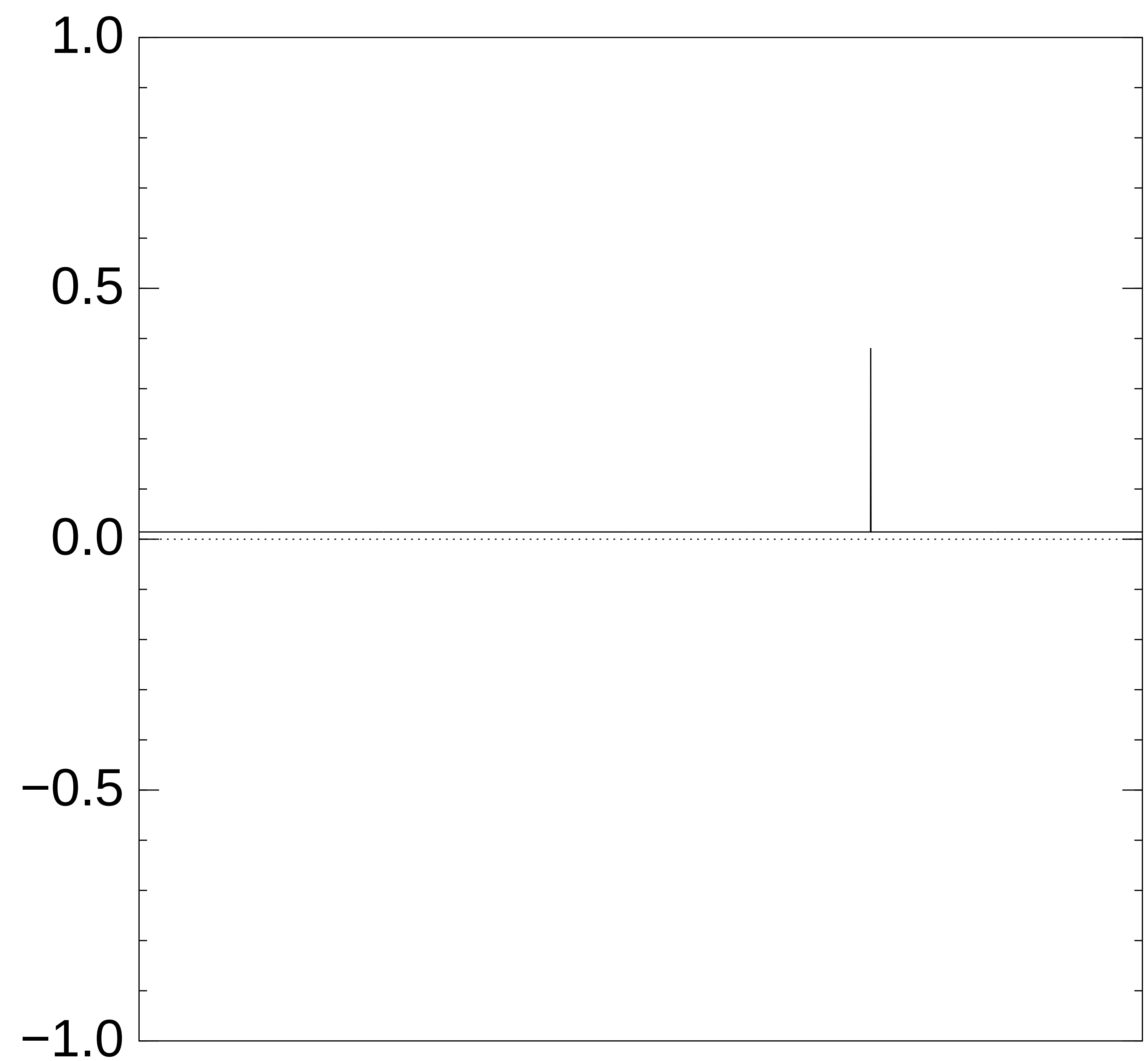
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $12 \times$ (Step $1 +$ Step 2):



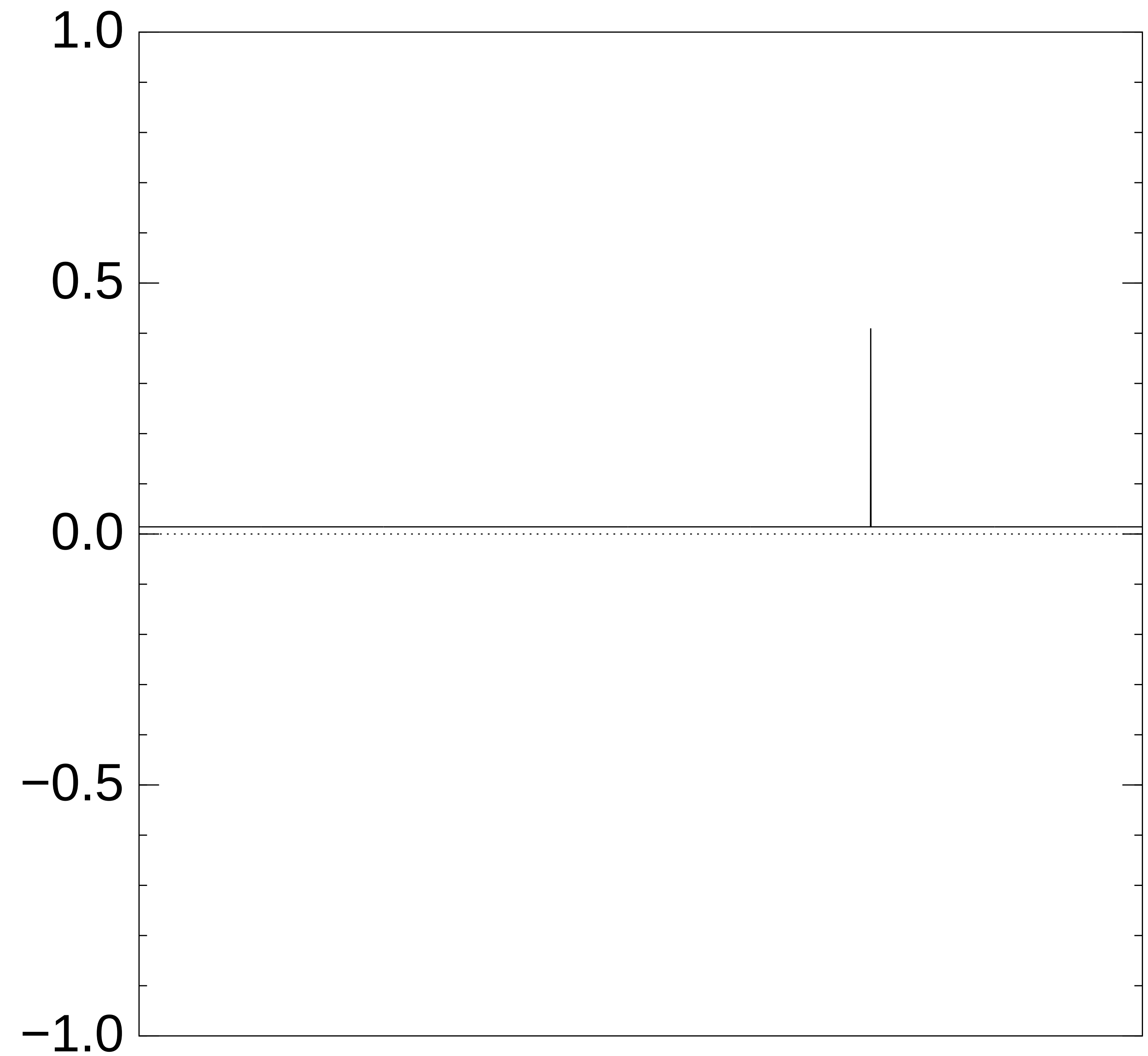
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $13 \times$ (Step 1 + Step 2):



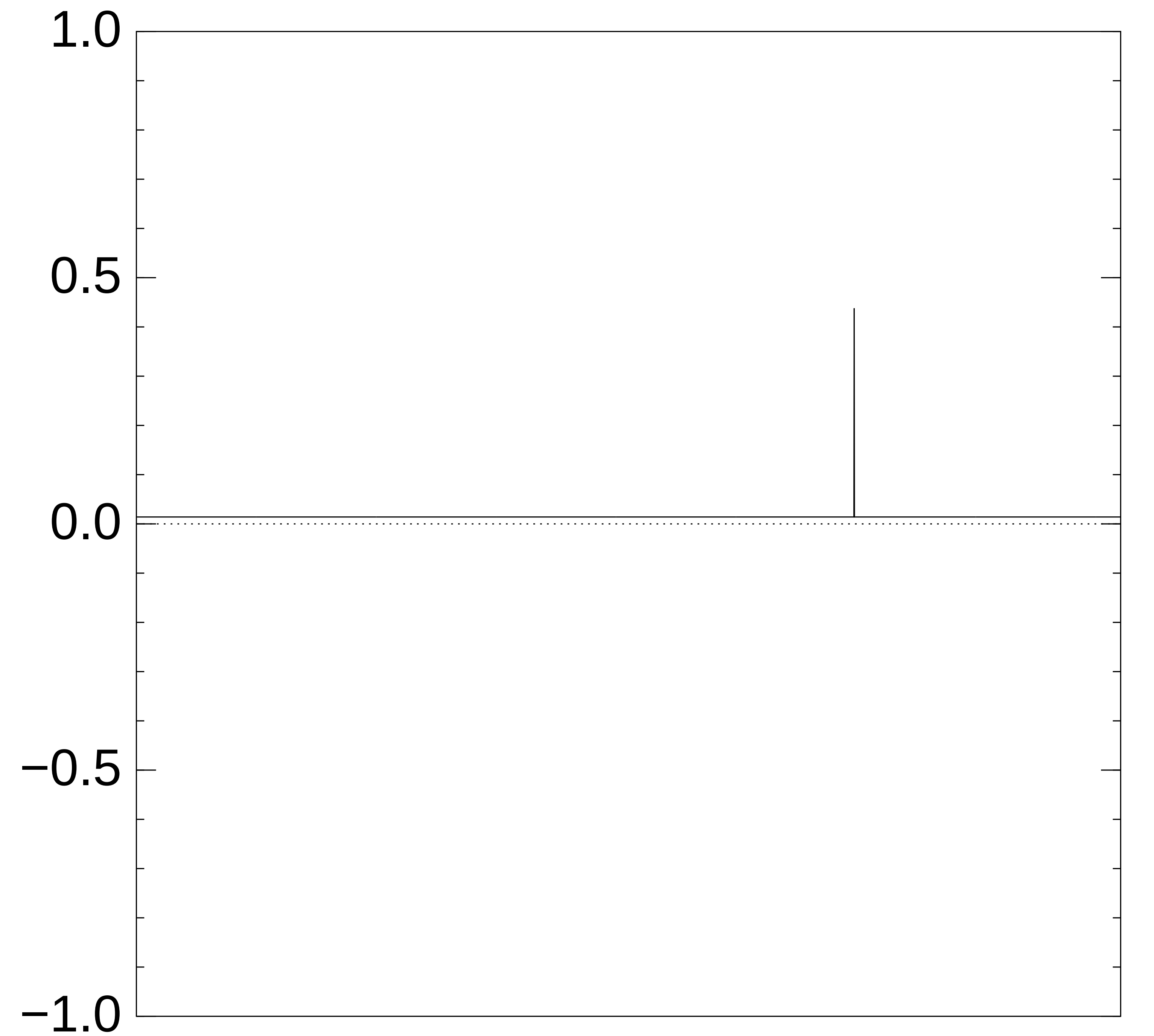
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $14 \times$ (Step 1 + Step 2):



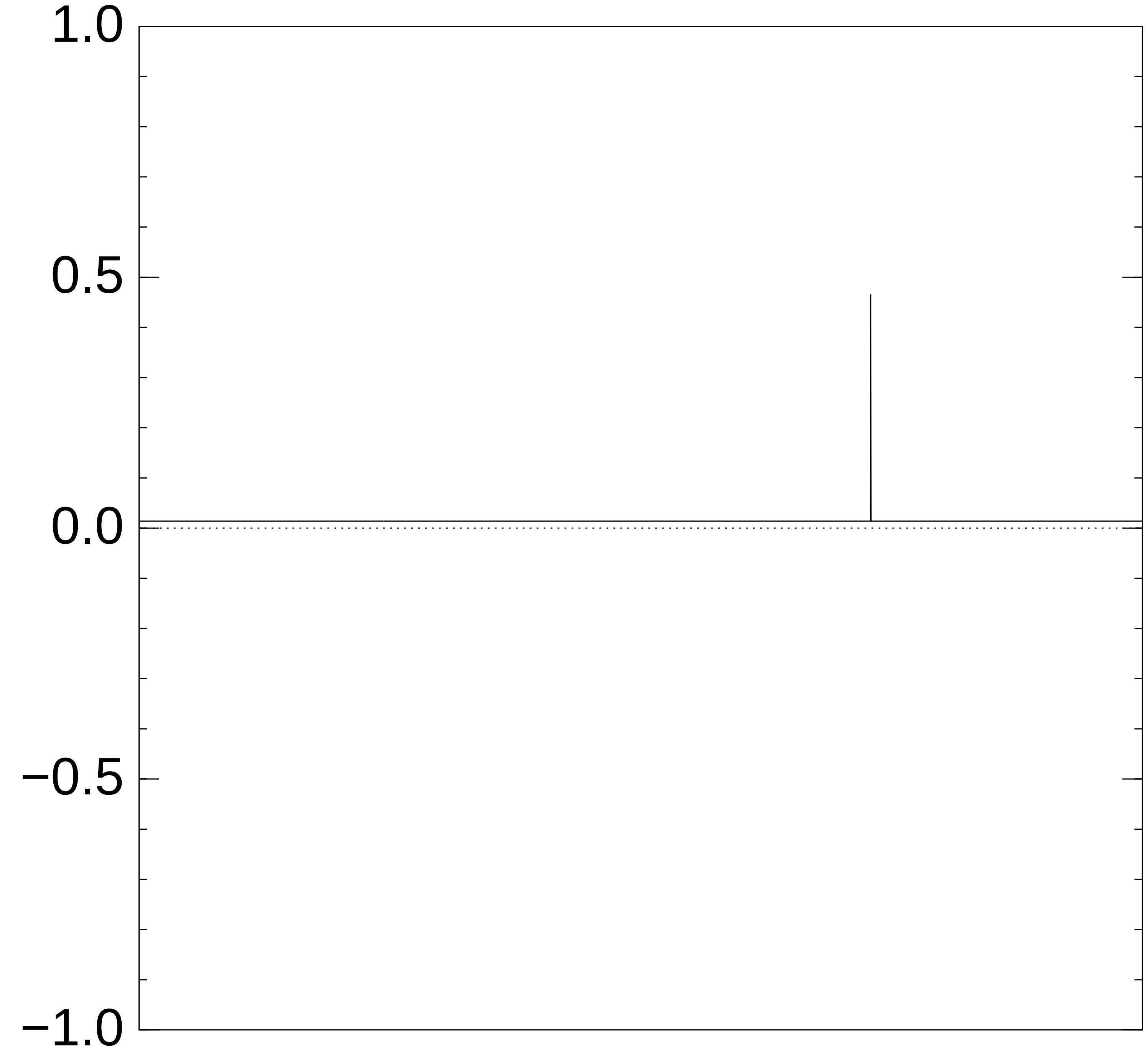
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $15 \times$ (Step 1 + Step 2):



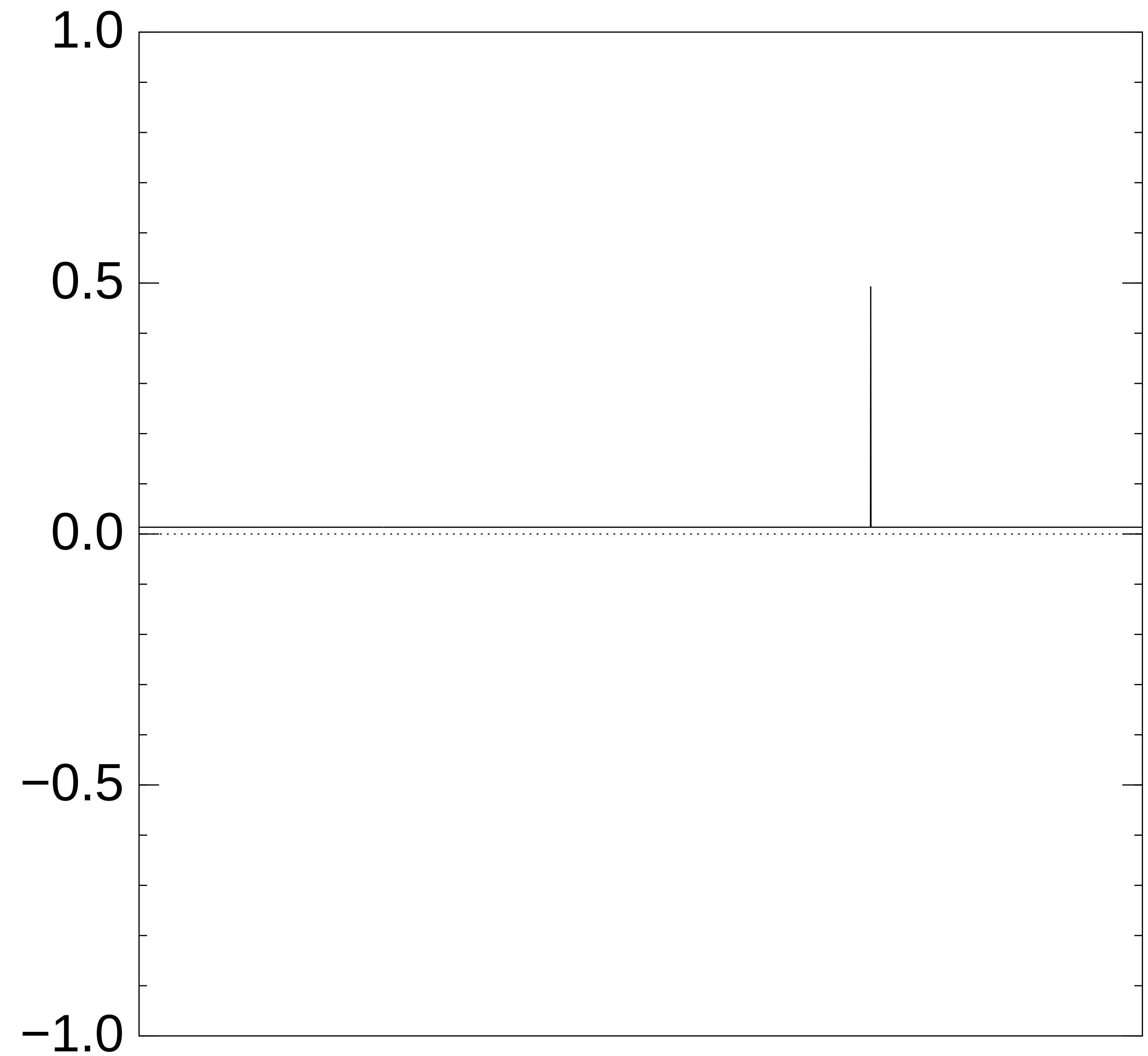
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $16 \times$ (Step 1 + Step 2):



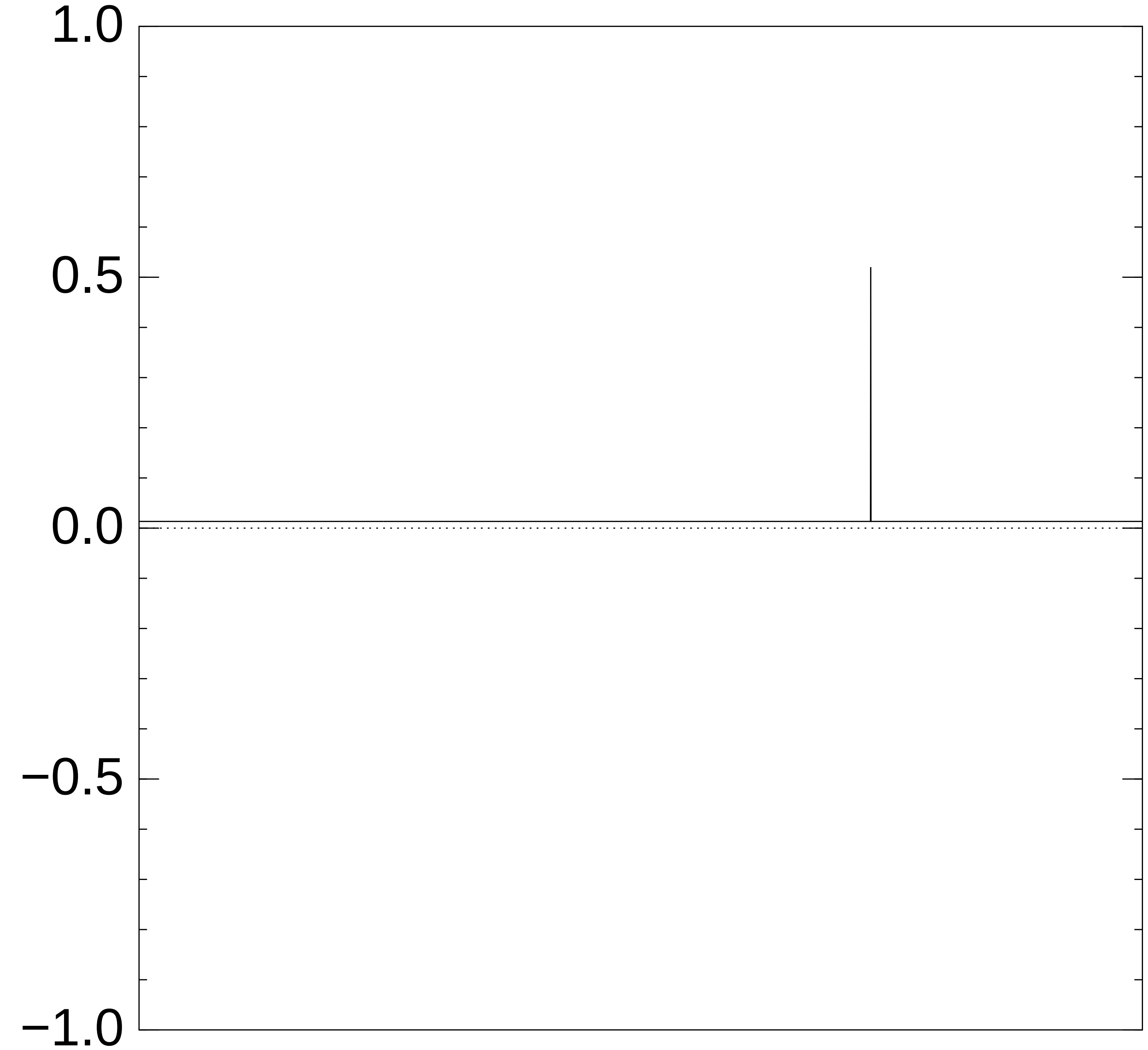
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $17 \times$ (Step 1 + Step 2):



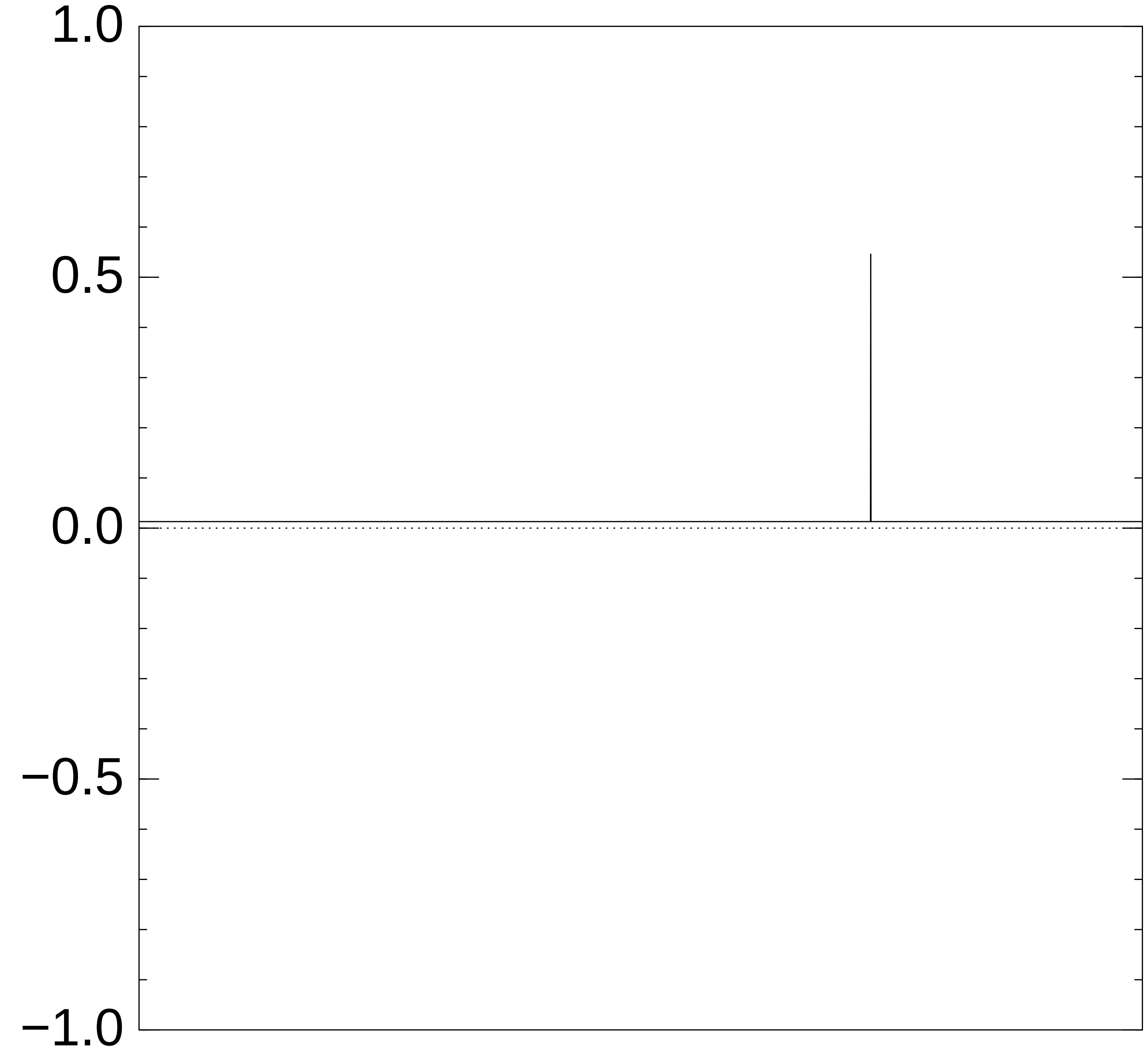
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n)\sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $18 \times$ (Step 1 + Step 2):



Step 1: Set $a \leftarrow b$ where

$b_J = -a_J$ if $\Sigma(J) = t$,

$b_J = a_J$ otherwise.

This is about as easy as computing Σ.

Step 2: "Grover diffusion".

Set $a \leftarrow b$ where

$b_J = -a_J + (2/2^n) \sum_I a_I$.

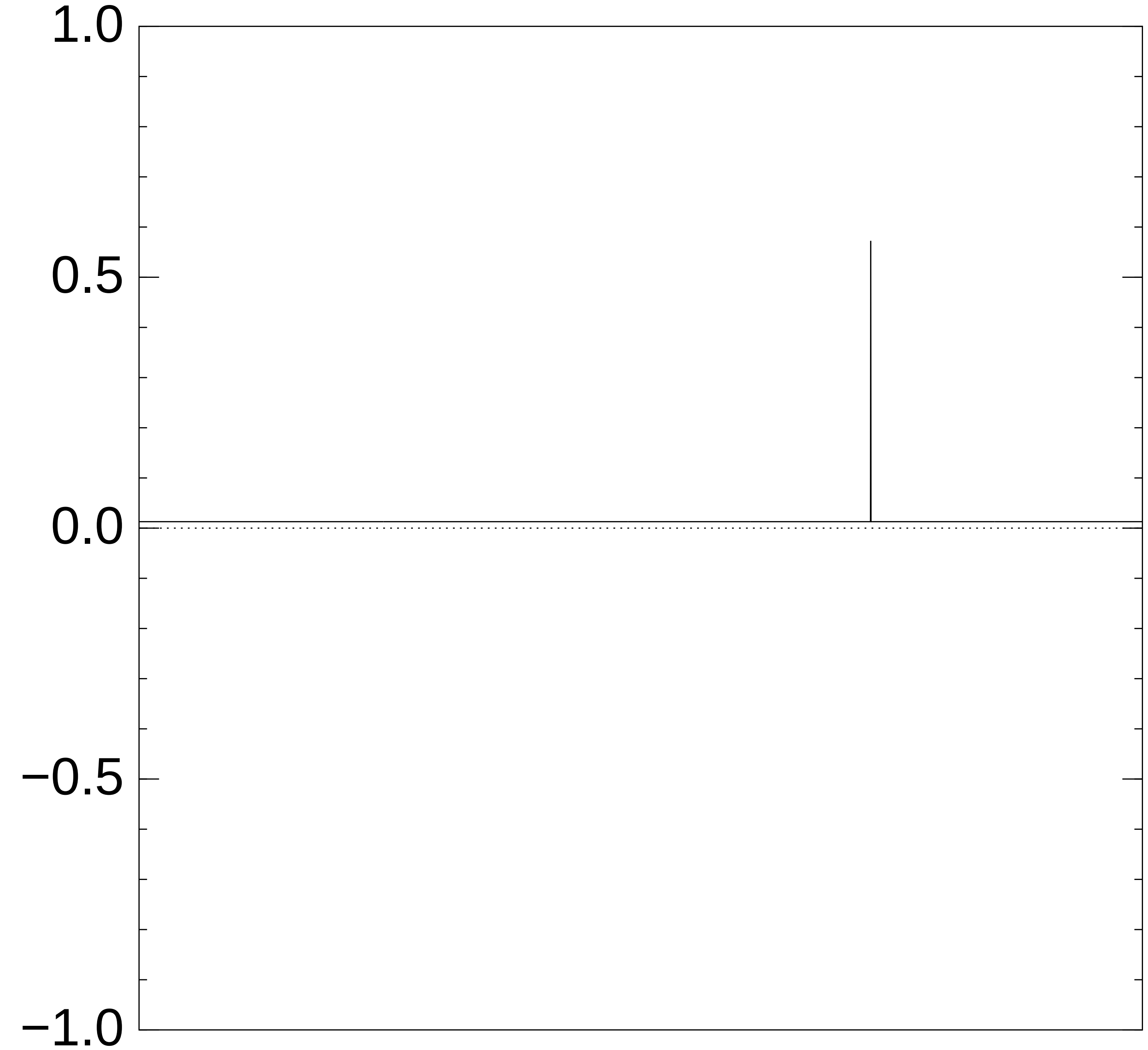
This is also easy.

Repeat steps 1 and 2 about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.

With high probability this finds the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$ for 36634 example with $n = 12$ after $19 \times$ (Step 1 + Step 2):



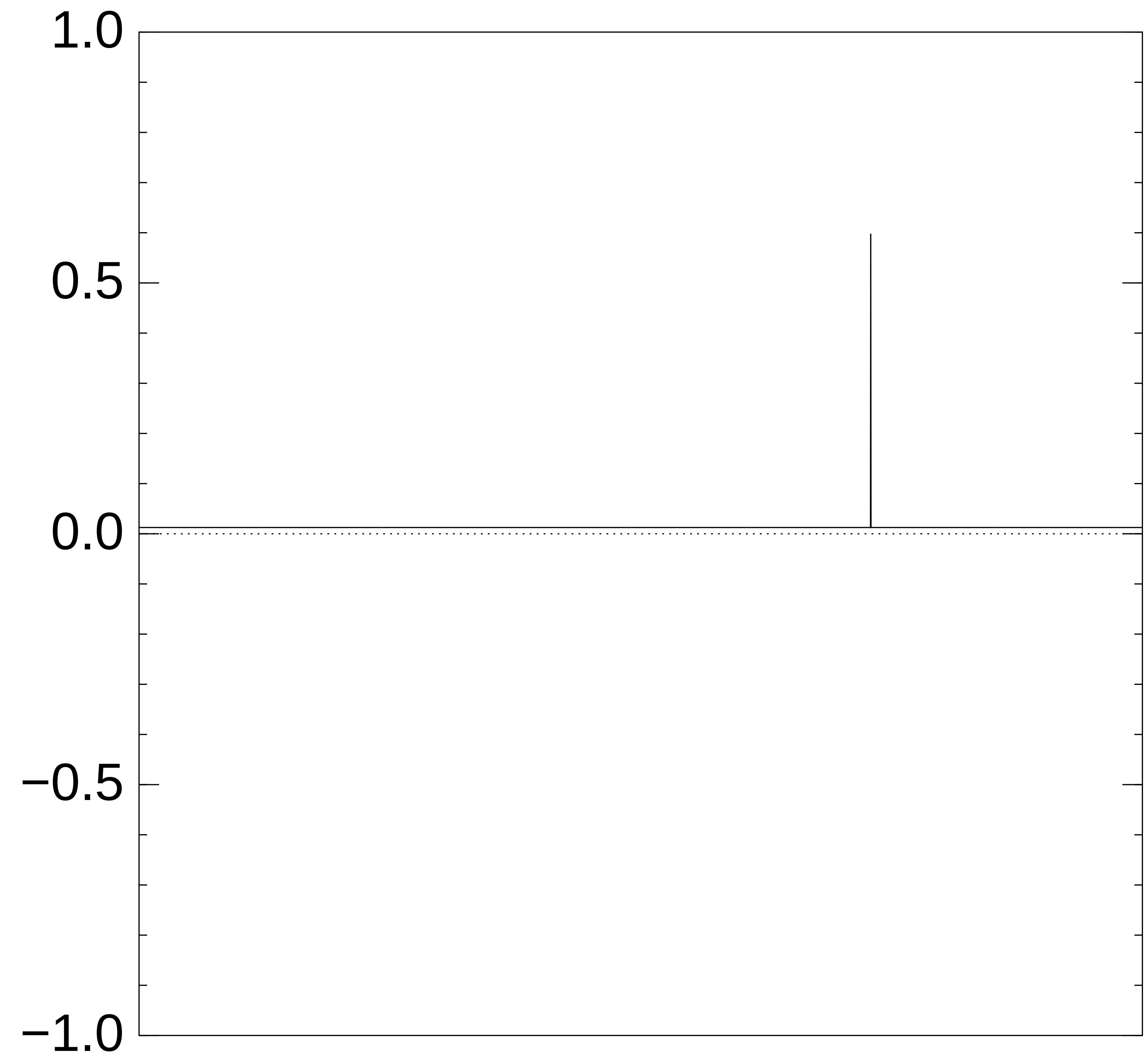
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $20 \times$ (Step 1 + Step 2):



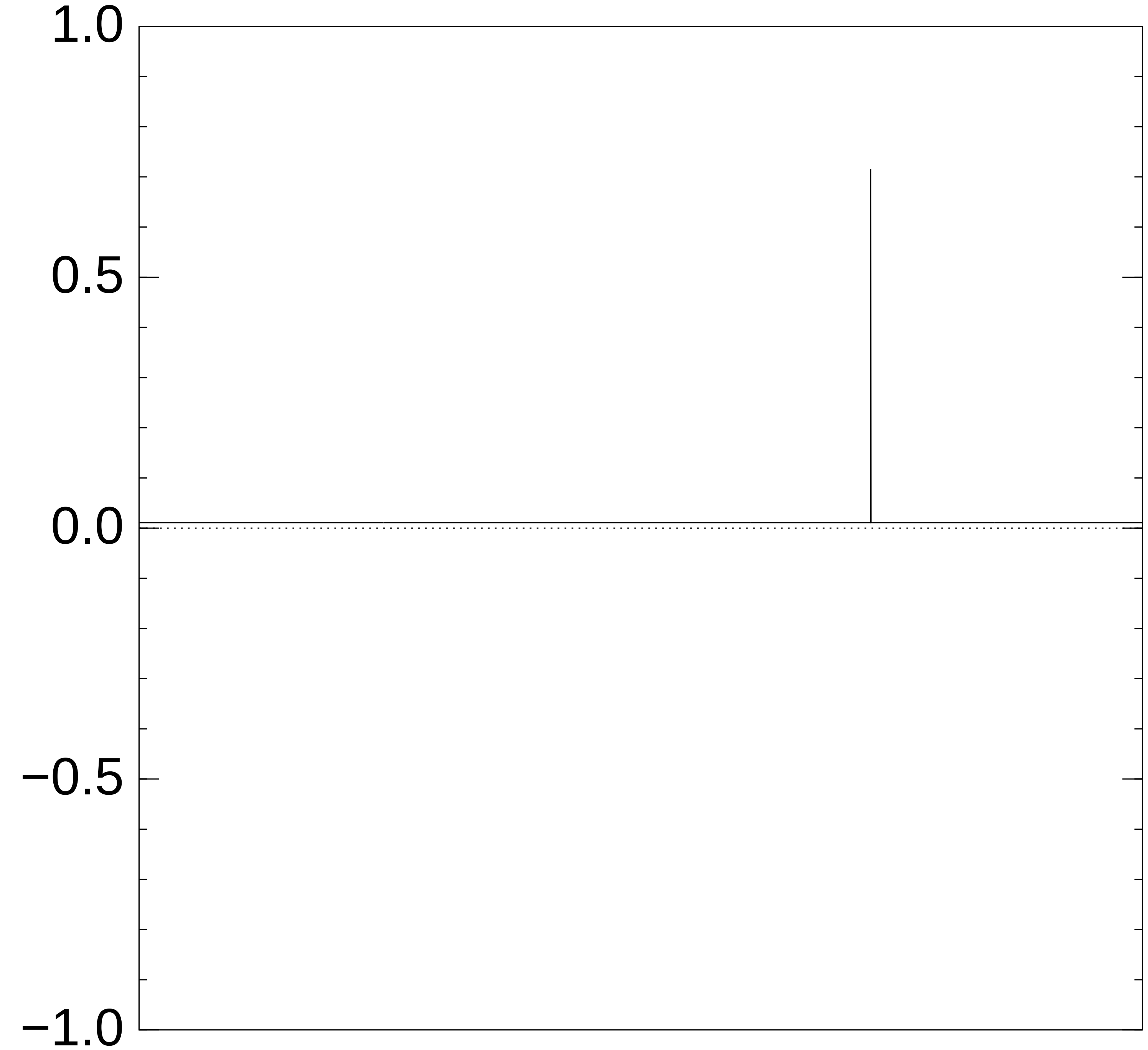
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n)\sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $25 \times$ (Step 1 + Step 2):



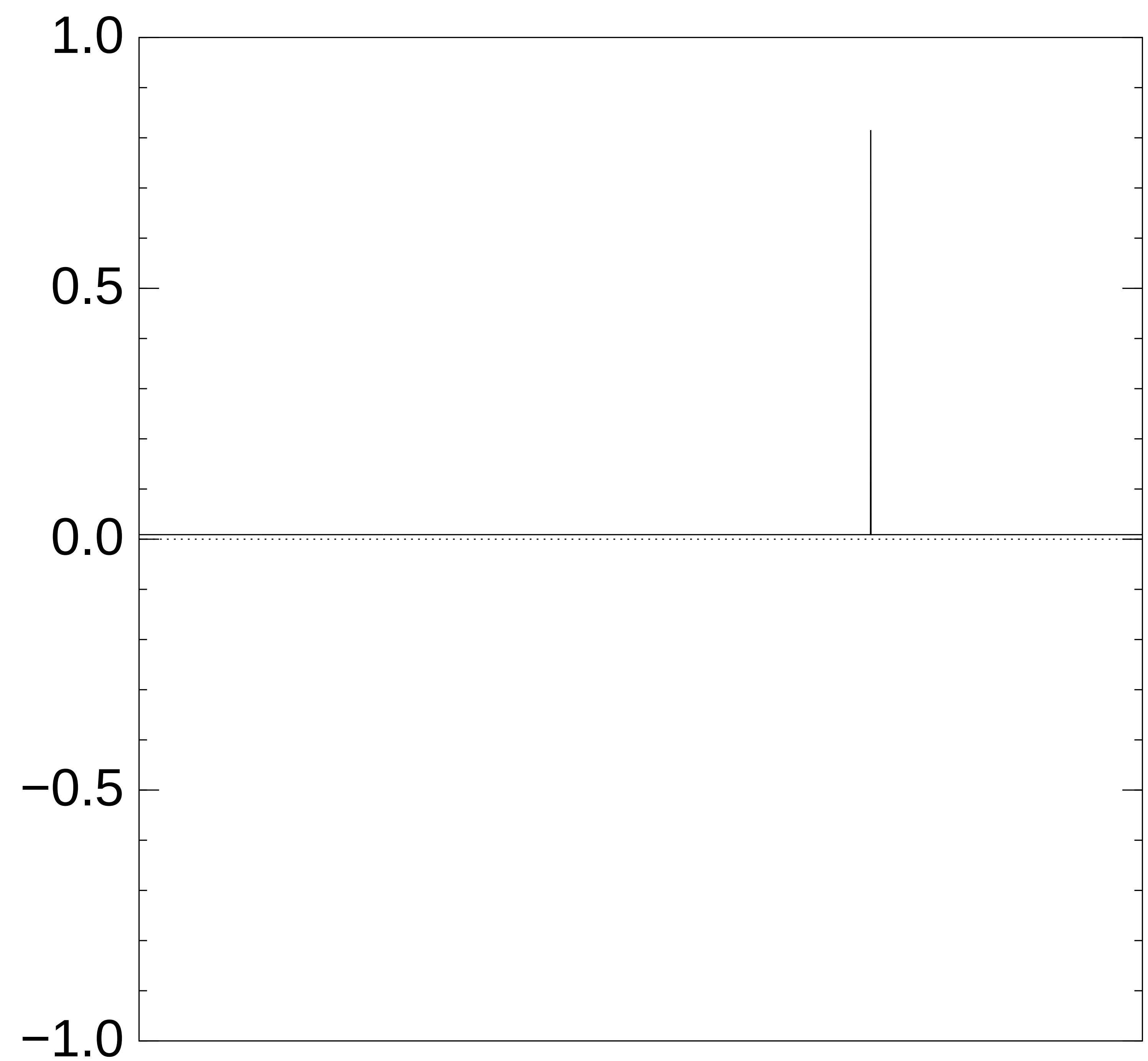
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $30 \times$ (Step 1 + Step 2):



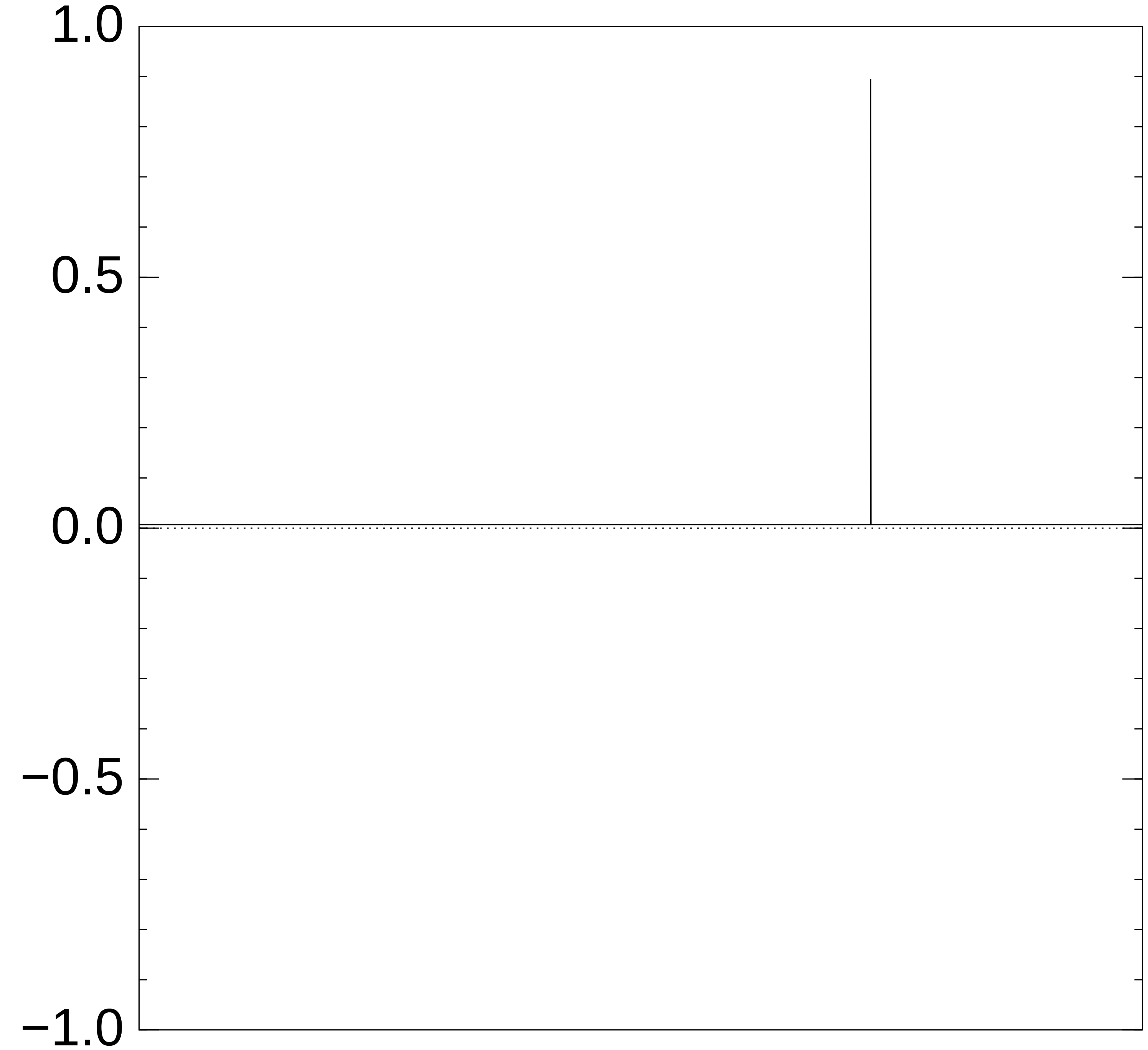
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $35 \times$ (Step 1 + Step 2):



Good moment to stop, measure.

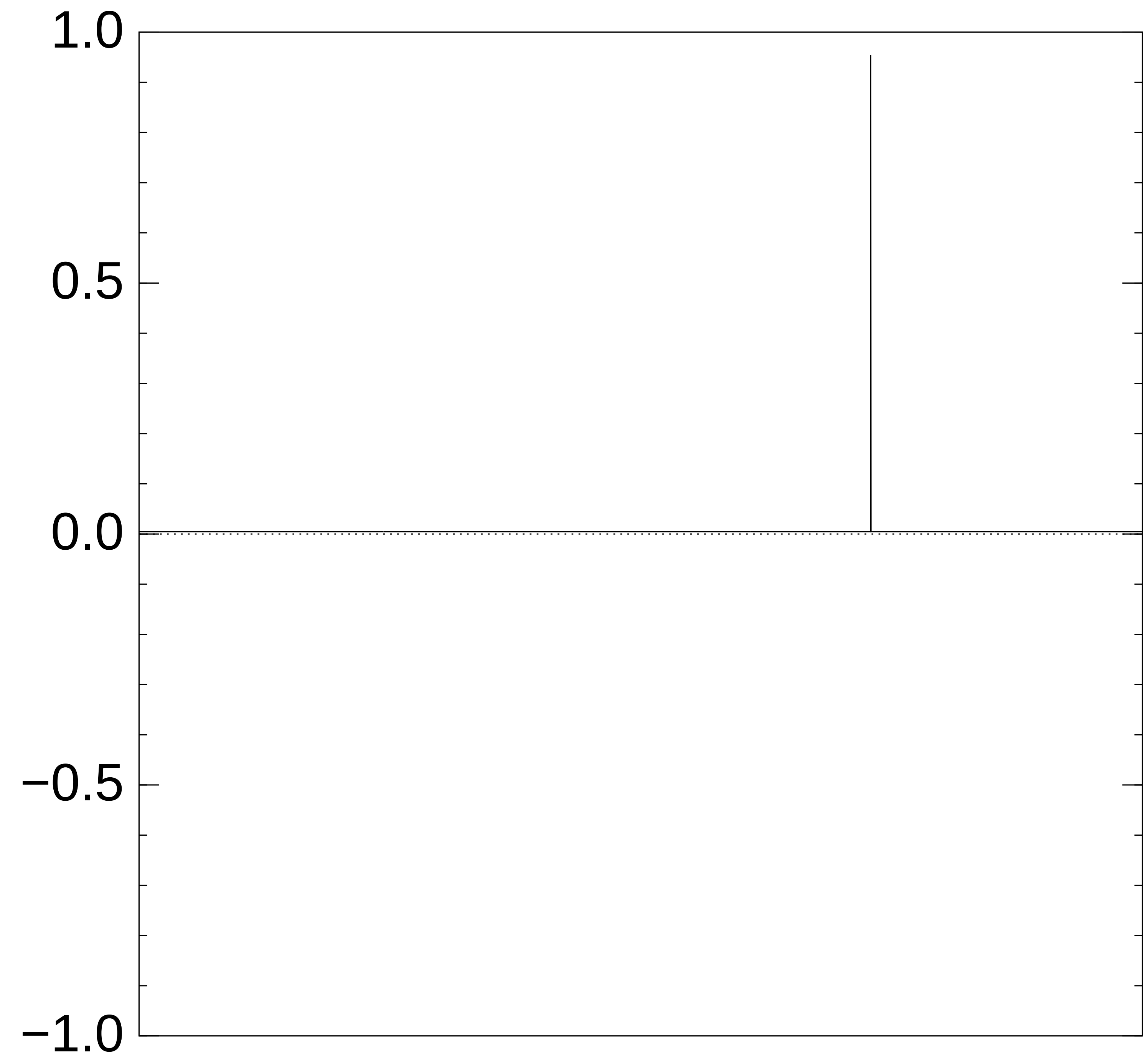
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $40 \times$ (Step 1 + Step 2):



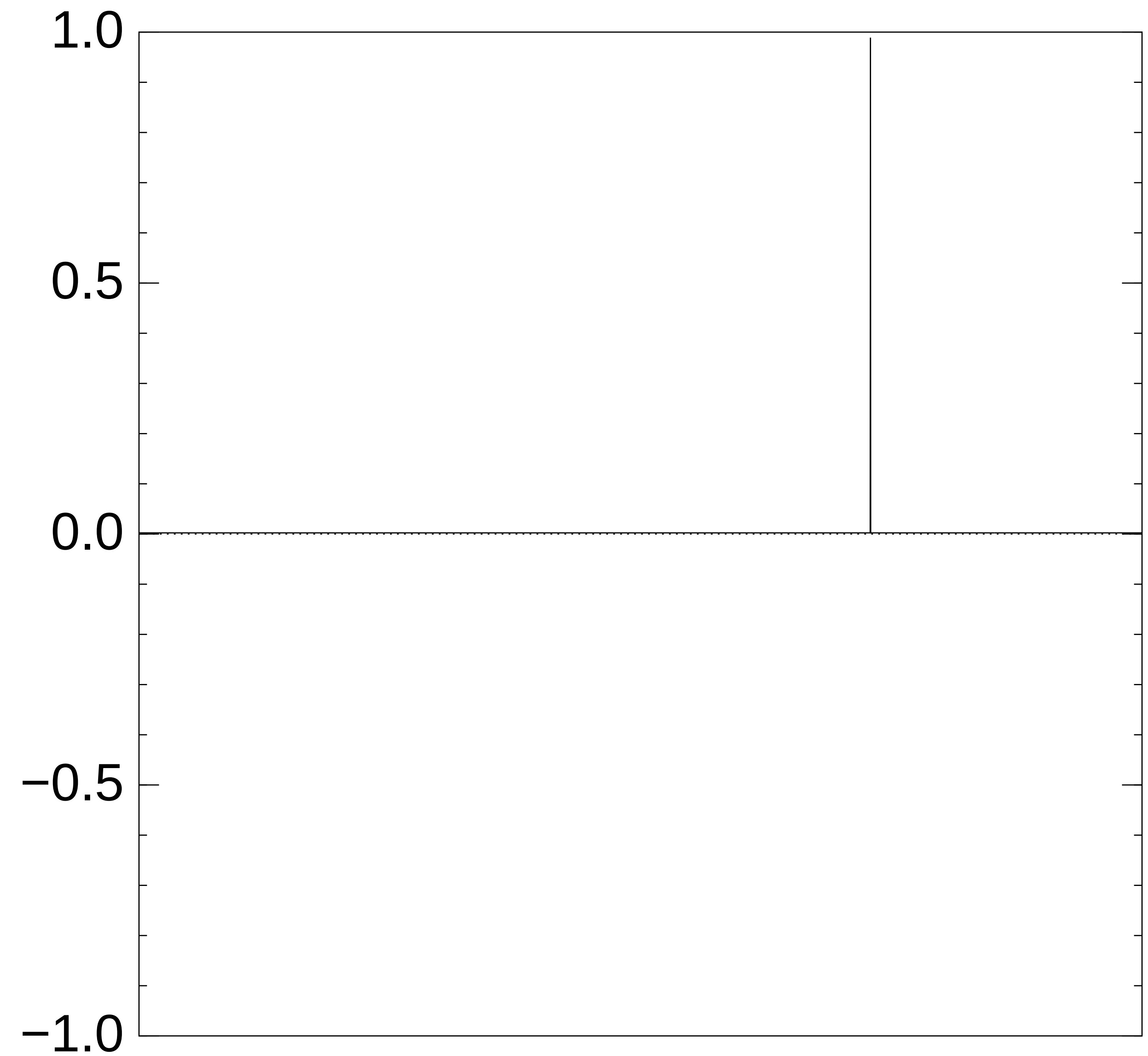
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2 about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$ for 36634 example with $n = 12$ after $45 \times$ (Step 1 + Step 2):



Step 1: Set $a \leftarrow b$ where

$b_J = -a_J$ if $\Sigma(J) = t$,

$b_J = a_J$ otherwise.

This is about as easy as computing Σ.

Step 2: "Grover diffusion".

Set $a \leftarrow b$ where

$b_J = -a_J + (2/2^n) \sum_I a_I$.

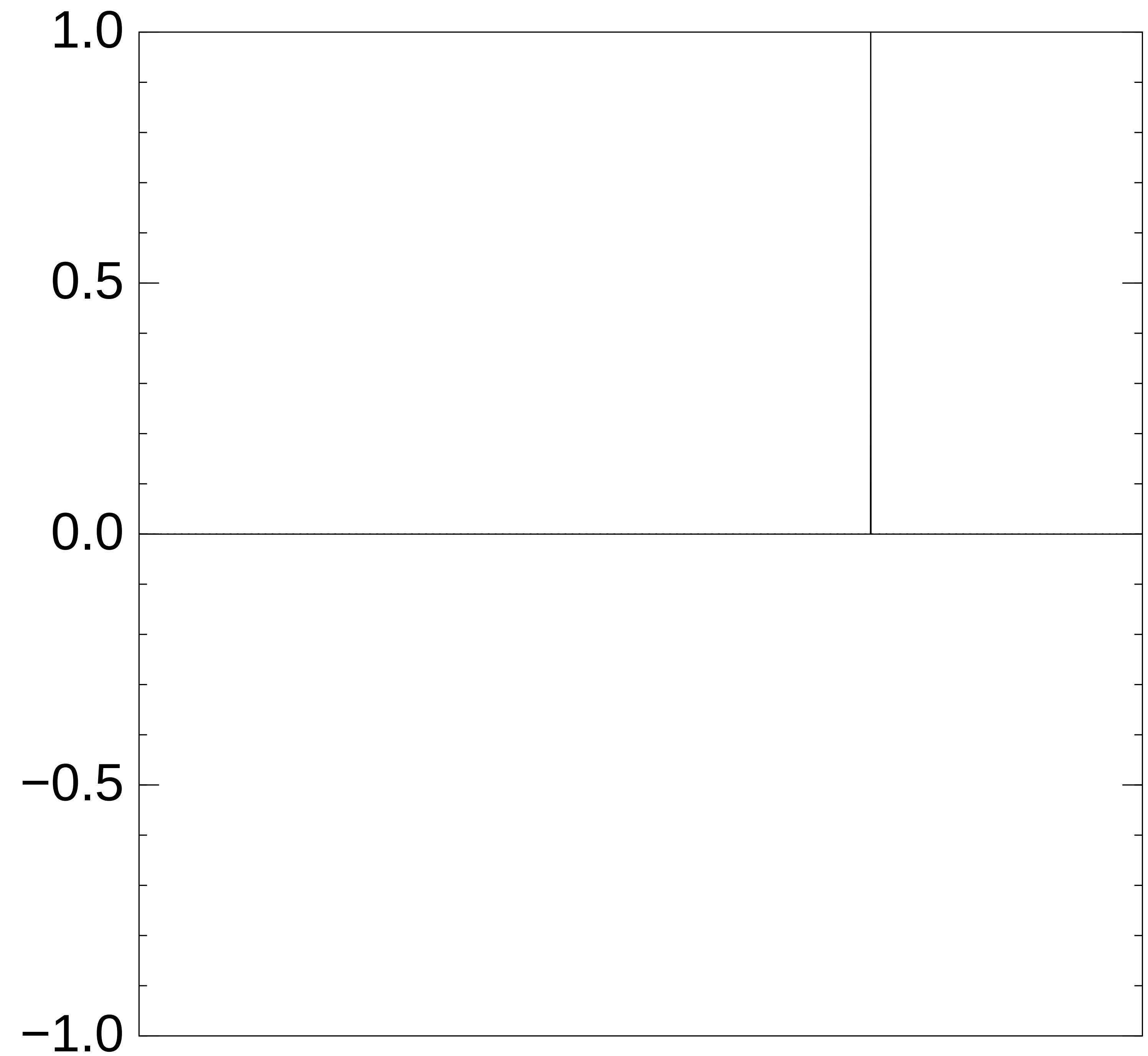
This is also easy.

Repeat steps 1 and 2 about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.

With high probability this finds the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$ for 36634 example with $n = 12$ after $50 \times$ (Step 1 + Step 2):



Traditional stopping point.

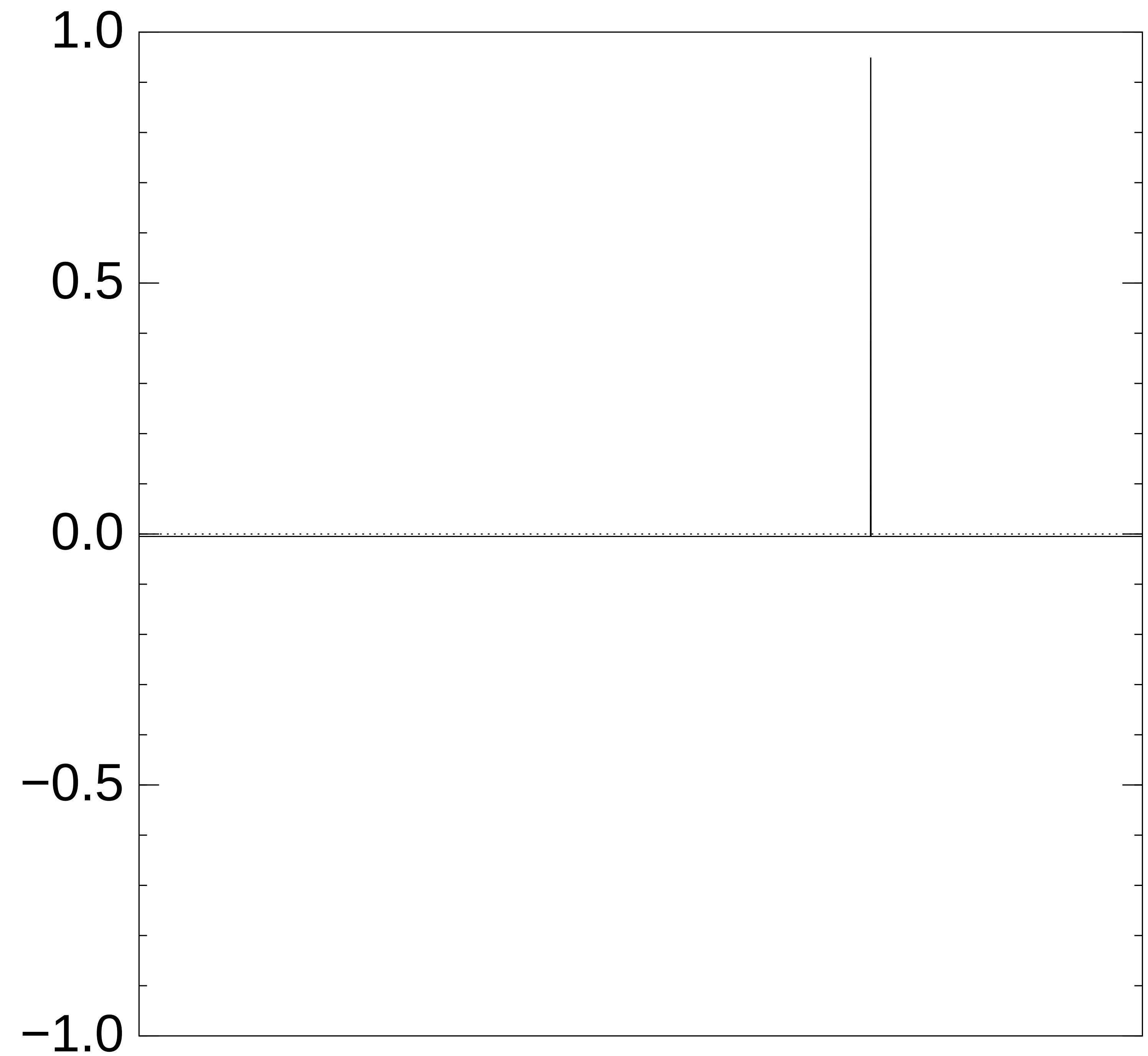
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $60 \times$ (Step 1 + Step 2):



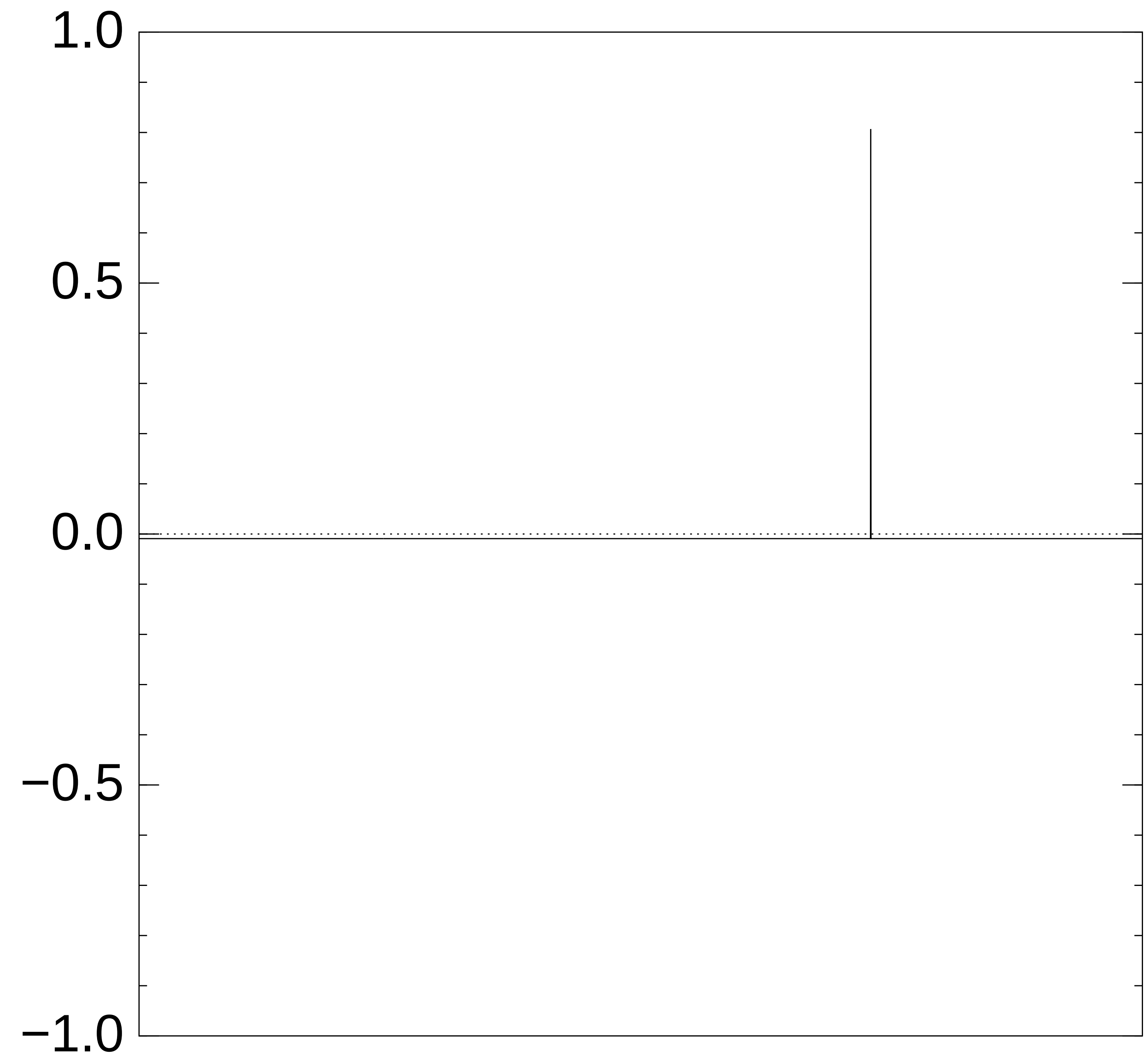
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n)\sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $70 \times$ (Step 1 + Step 2):



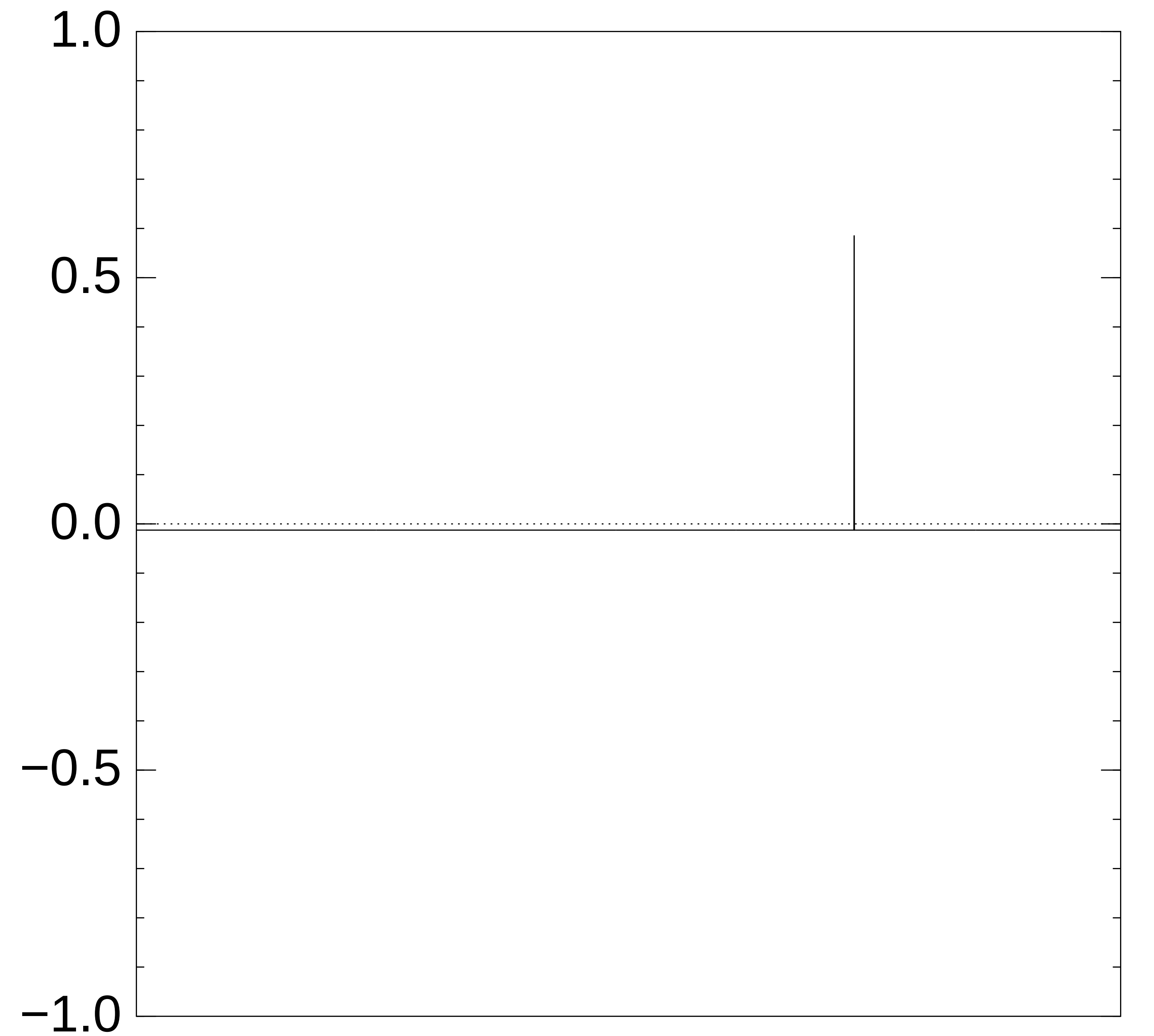
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $80 \times$ (Step 1 + Step 2):



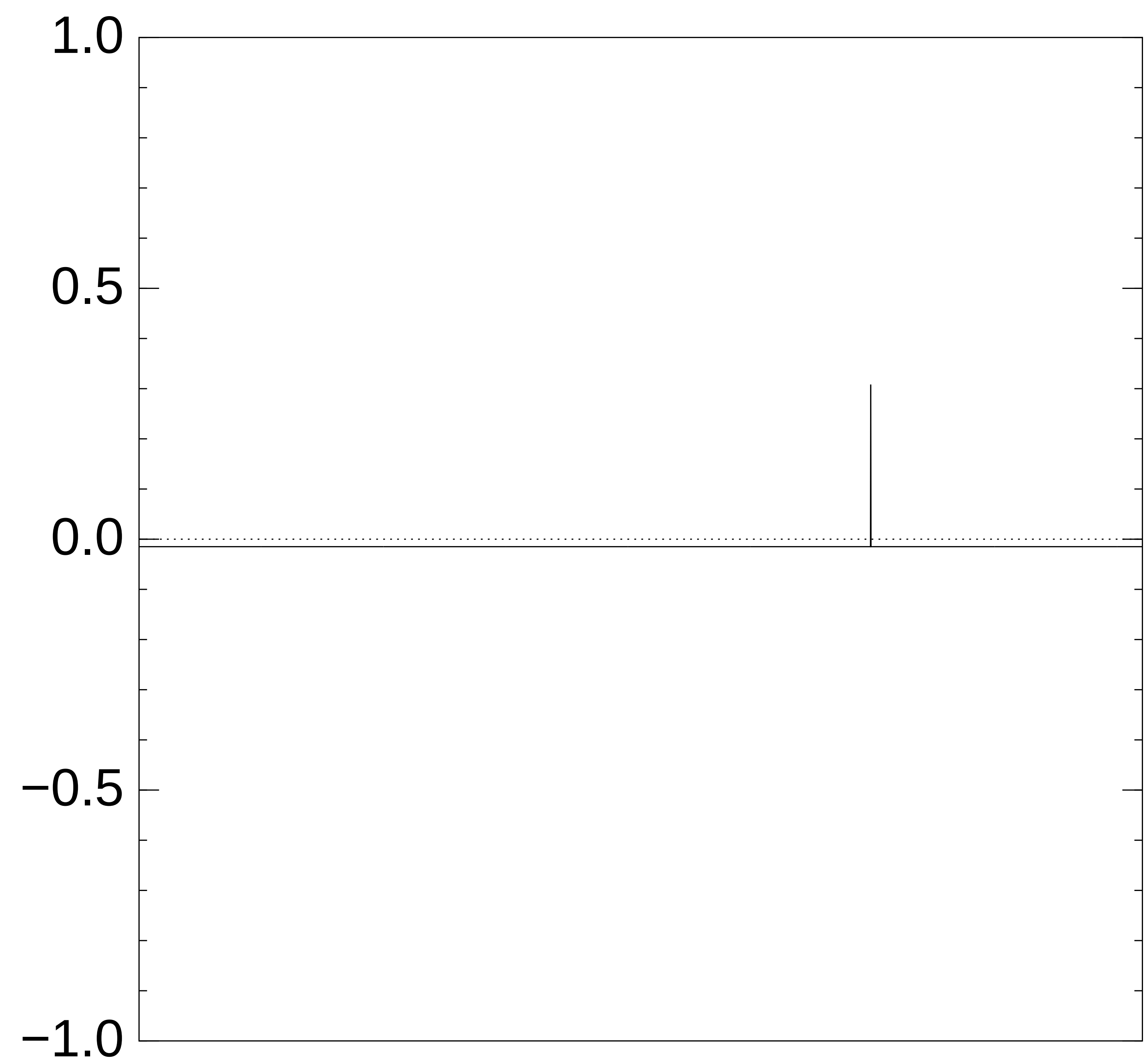
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $90 \times$ (Step 1 + Step 2):



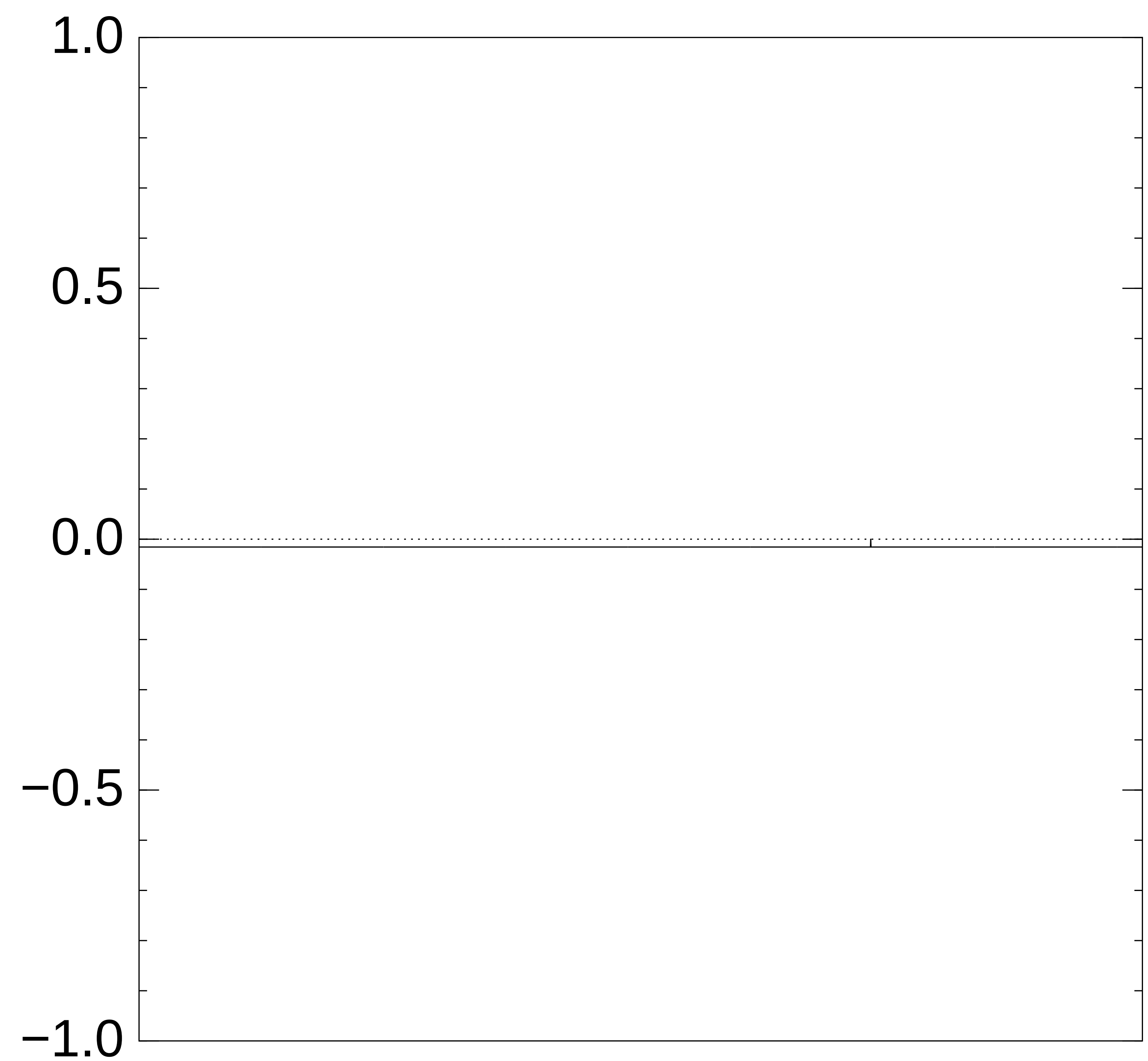
Step 1: Set $a \leftarrow b$ where
$b_J = -a_J$ if $\Sigma(J) = t$,
$b_J = a_J$ otherwise.
This is about as easy
as computing Σ.

Step 2: "Grover diffusion".
Set $a \leftarrow b$ where
$b_J = -a_J + (2/2^n) \sum_I a_I$.
This is also easy.

Repeat steps 1 and 2
about $0.58 \cdot 2^{0.5n}$ times.

Measure the n qubits.
With high probability this finds
the unique J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $100 \times$ (Step 1 + Step 2):



Very bad stopping point.

Set $a \leftarrow b$ where

$_J$ if $\Sigma(J) = t$,

otherwise.

bout as easy

uting Σ.

"Grover diffusion".

b where

$_J + (2/2^n)\sum_I a_I$.

lso easy.

steps 1 and 2

$58 \cdot 2^{0.5n}$ times.

the n qubits.

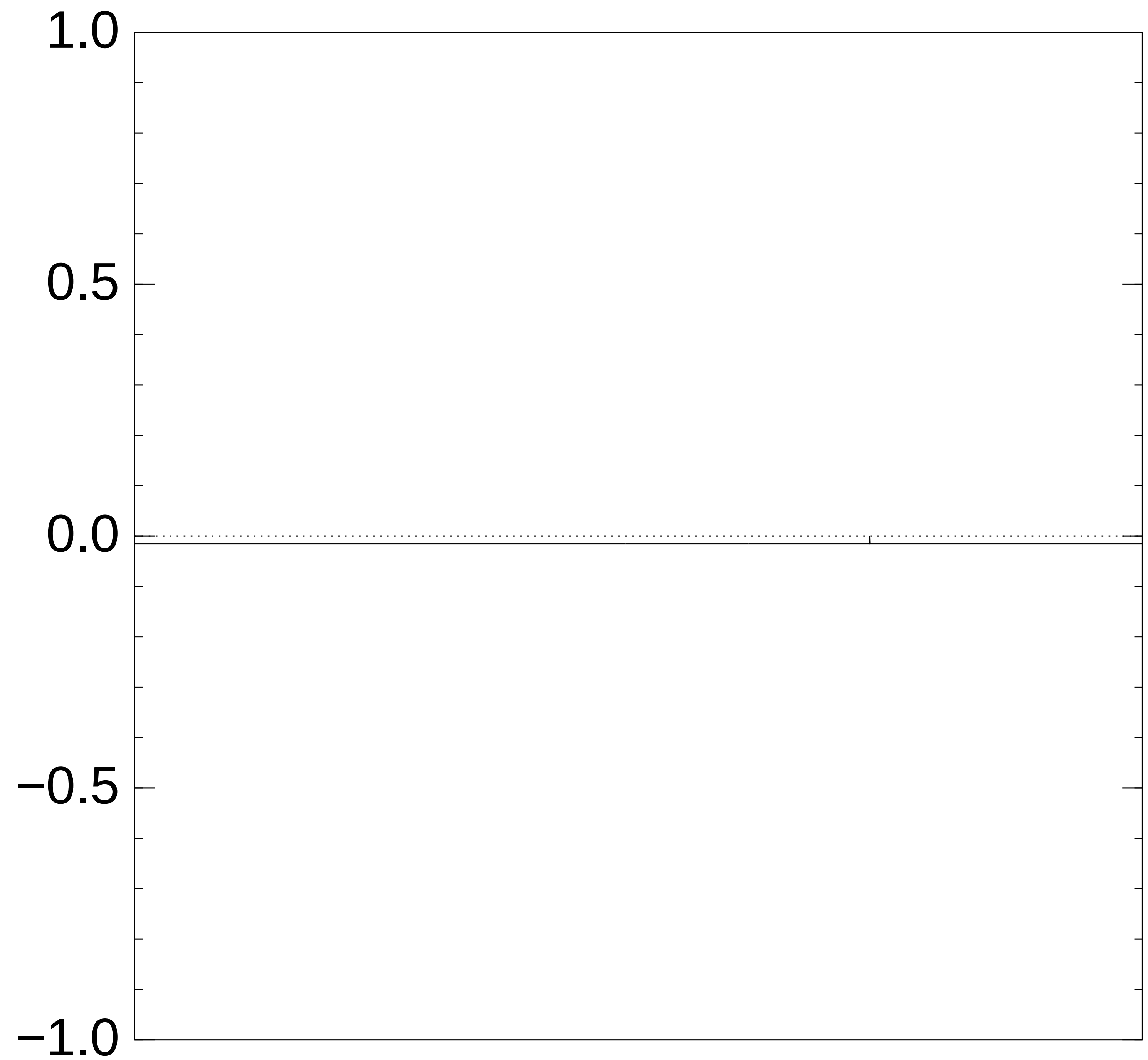
gh probability this finds

ue J such that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$

for 36634 example with $n = 12$

after $100 \times$ (Step 1 + Step 2):



Very bad stopping point.

$J \mapsto a_J$

by a vec

(with fix

(1) a_J f

(2) a_J f

Step 1 +

act linea

Easily co

and pow

to under

of state

$\Rightarrow$ Proba

after $\approx$(

where
$= t$,
.
asy

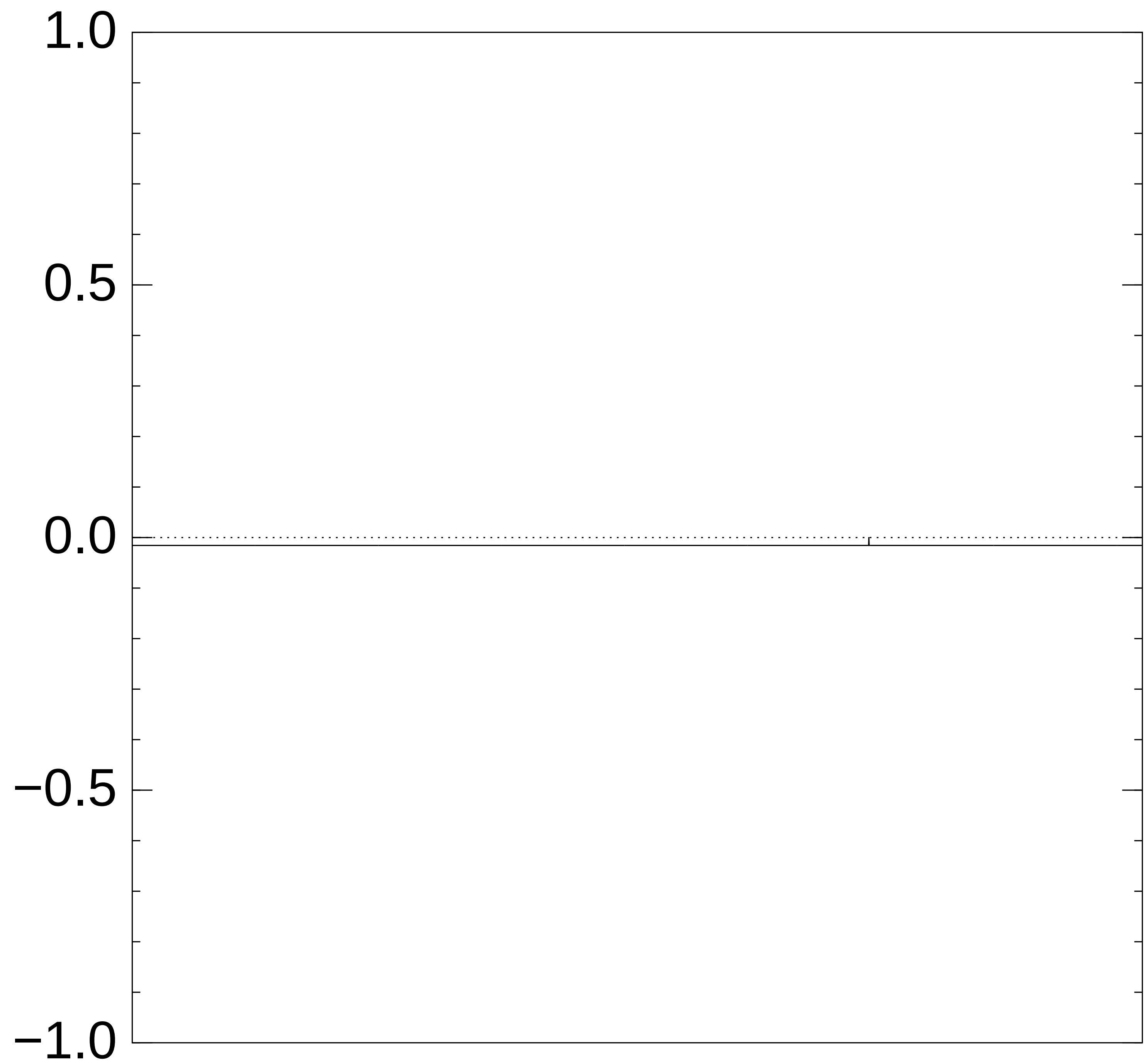
iffusion".

$) \sum_I a_I$.

d 2
times.

bits.
lity this finds
that $\Sigma(J) = t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $100 \times$ (Step 1 + Step 2):



Very bad stopping point.

$J \mapsto a_J$ is complet
by a vector of two
(with fixed multipl
(1) a_J for roots J
(2) a_J for non-roo

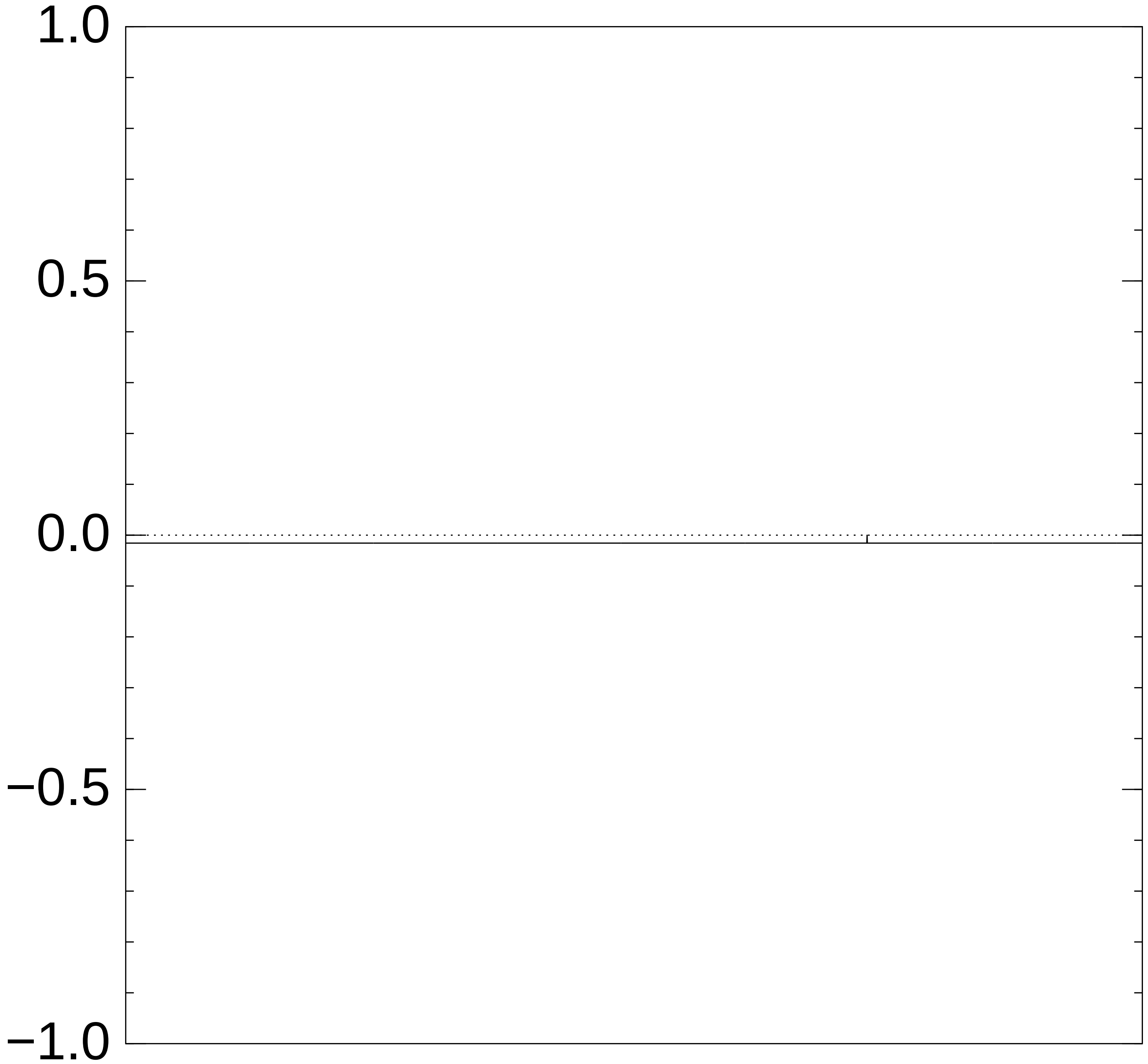
Step 1 + Step 2
act linearly on this

Easily compute eig
and powers of this
to understand evo
of state of Grover'
$\Rightarrow$ Probability is $\approx$
after $\approx(\pi/4)2^{0.5n}$

nds

) $= t$.

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $100 \times (\text{Step 1} + \text{Step 2})$:



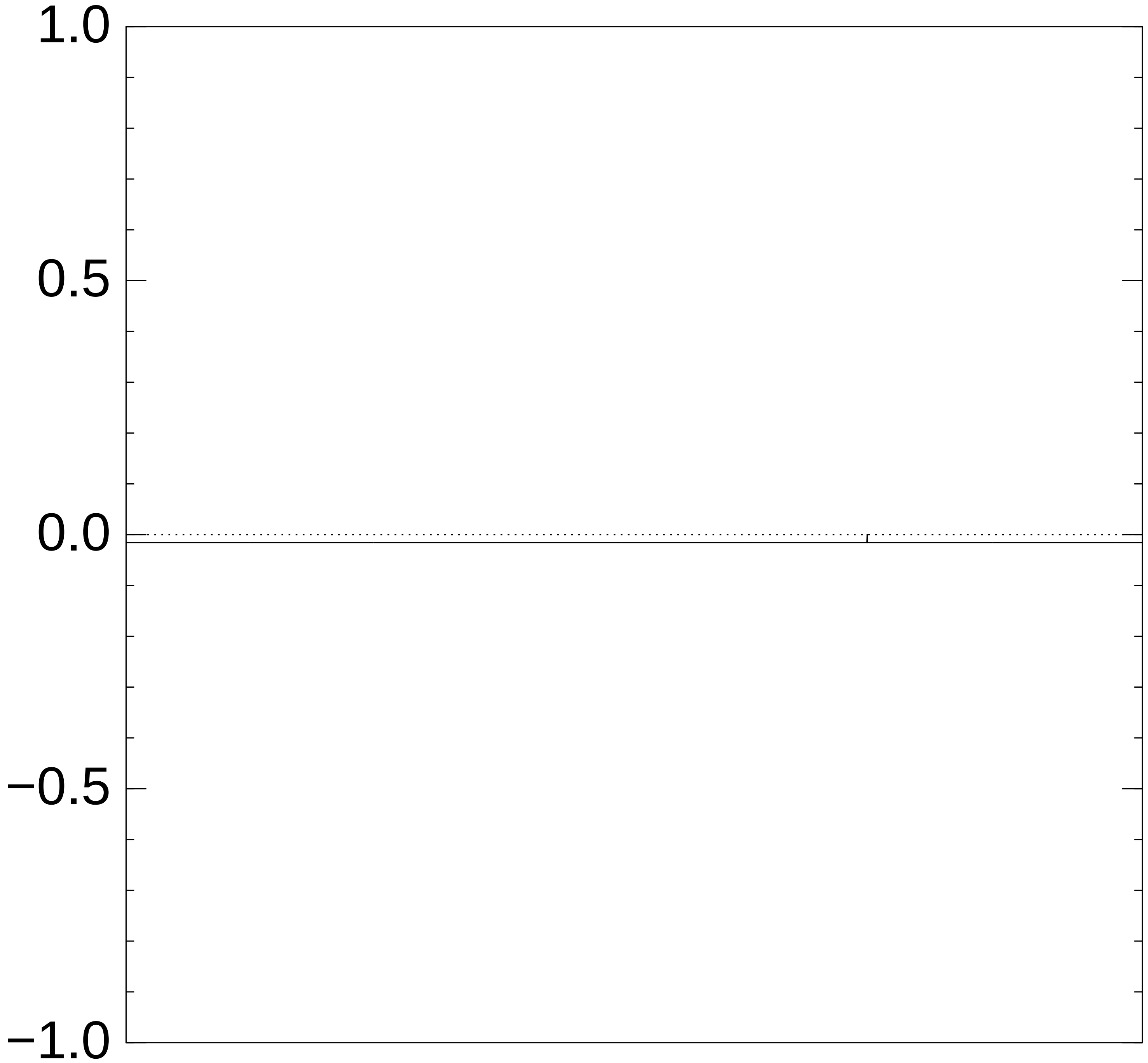
Very bad stopping point.

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

Step 1 + Step 2
act linearly on this vector.

Easily compute eigenvalues
and powers of this linear ma
to understand evolution
of state of Grover's algorithm
$\Rightarrow$ Probability is ≈ 1
after $\approx (\pi/4) 2^{0.5n}$ iterations

Graph of $J \mapsto a_J$
for 36634 example with $n = 12$
after $100 \times (\text{Step } 1 + \text{Step } 2)$:



Very bad stopping point.

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

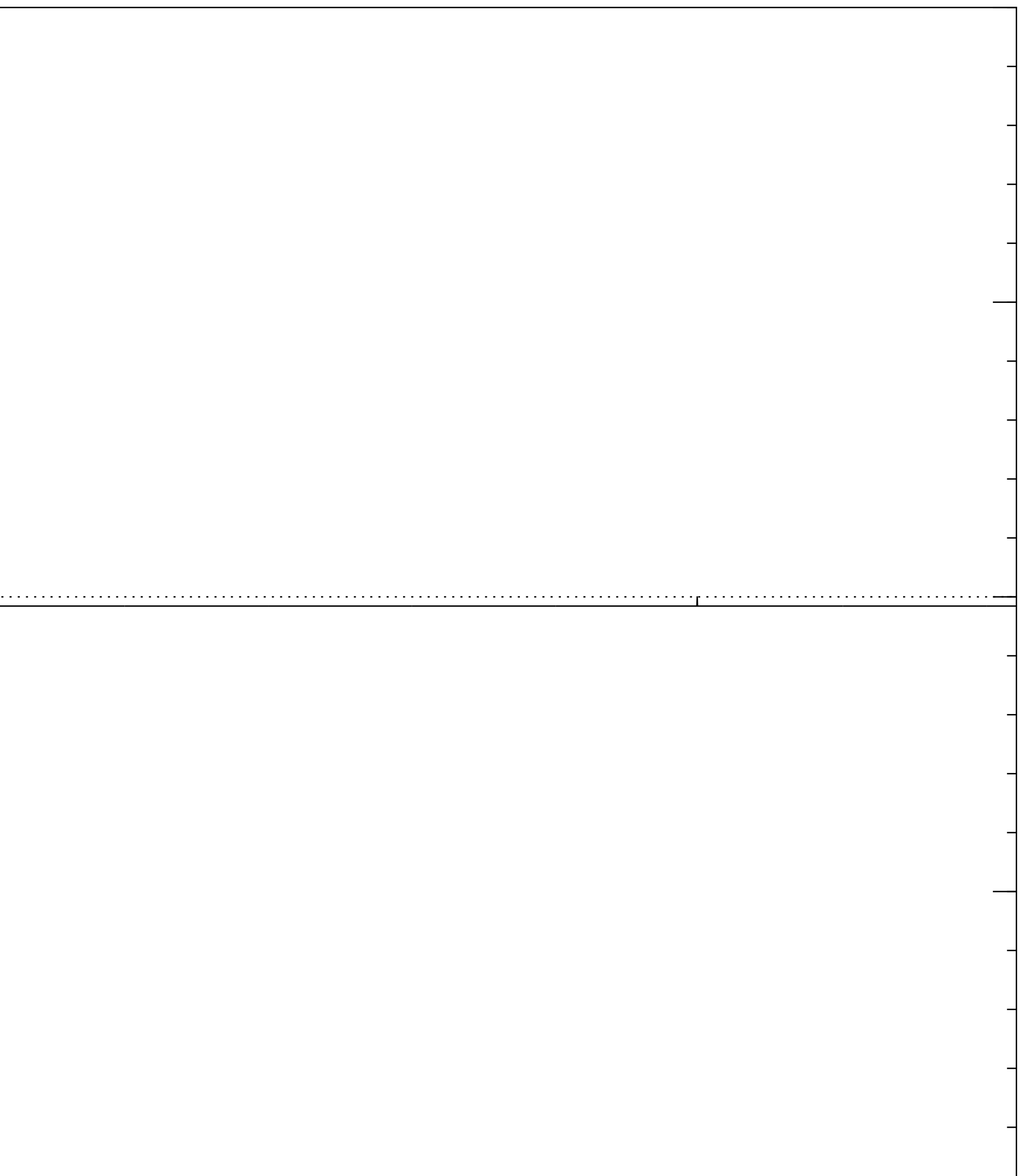
Step $1 +$ Step 2
act linearly on this vector.

Easily compute eigenvalues
and powers of this linear map
to understand evolution
of state of Grover's algorithm.
$\Rightarrow$ Probability is ≈ 1
after $\approx(\pi/4)2^{0.5n}$ iterations.

f $J \mapsto a_J$

4 example with $n = 12$

0 $\times$ (Step 1 + Step 2):



d stopping point.

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

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 $\approx(\pi/4)2^{0.5n}$ iterations.

Left-righ

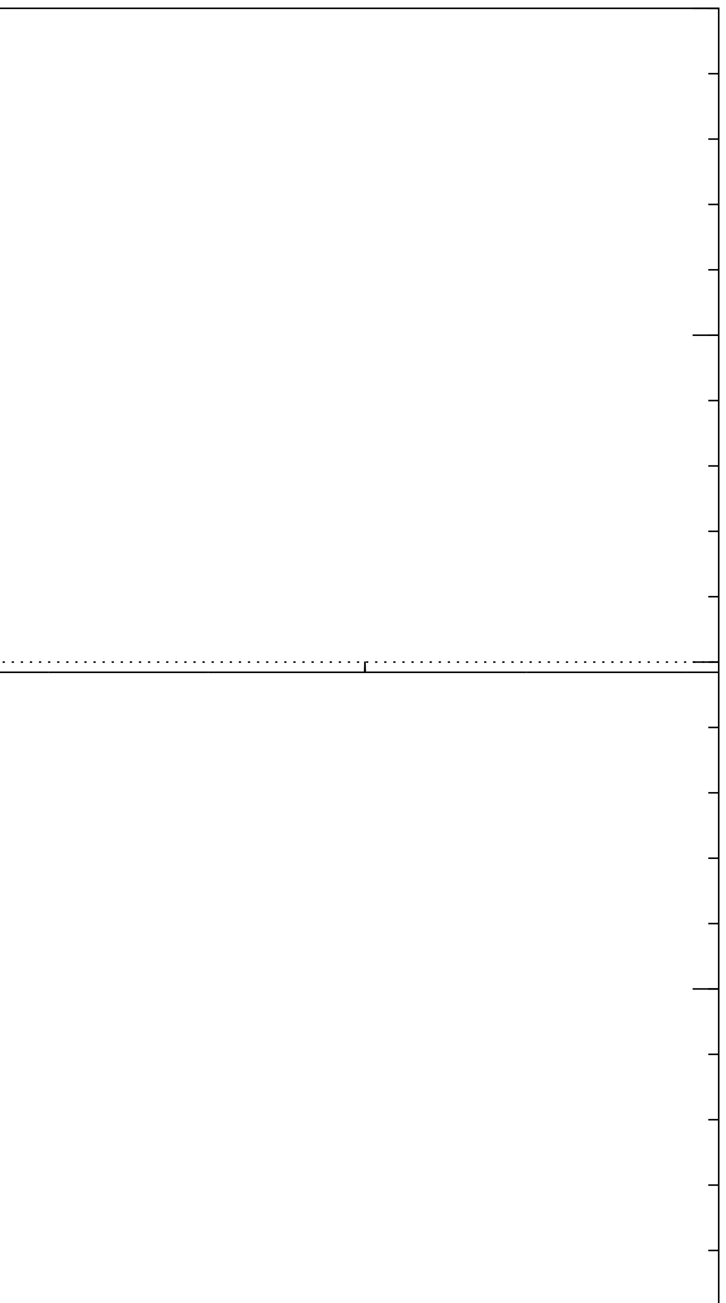
Don't ne
to achie

For simp

1974 Ho
Sort list
for all J_1
and list
for all J_2
Merge to
$\Sigma(J_1) =$
i.e., $\Sigma(J$

e with $n = 12$

1 + Step 2):



point.

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

(1) a_J for roots J;

(2) a_J for non-roots J.

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 $\approx(\pi/4)2^{0.5n}$ iterations.

<u>Left-right split (0.</u>

Don't need quantu

to achieve expone

For simplicity assu

1974 Horowitz–Sa

Sort list of $\Sigma(J_1)$

for all $J_1 \subseteq \{1, \ldots$

and list of $t - \Sigma($

for all $J_2 \subseteq \{n/2$

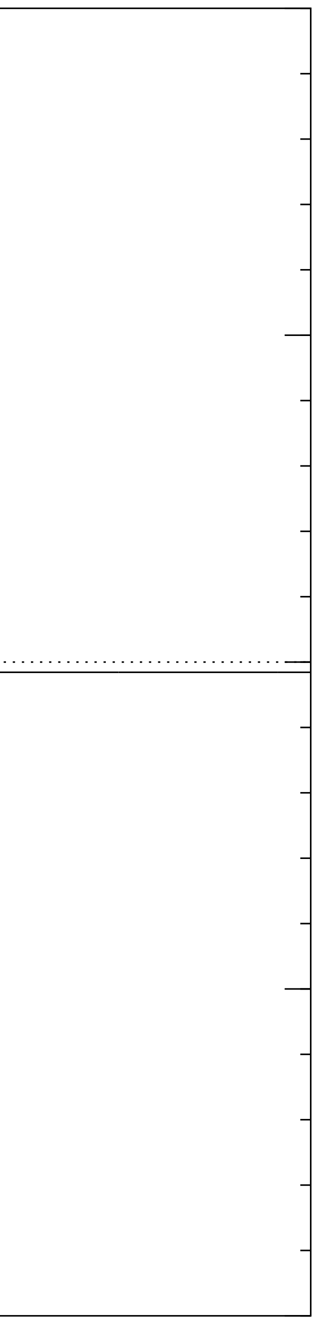
Merge to find colli

$\Sigma(J_1) = t - \Sigma(J_2$

i.e., $\Sigma(J_1 \cup J_2) =$

= 12

2):



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

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 $\approx(\pi/4)2^{0.5n}$ iterations.

Left-right split (0.5)

Don't need quantum compu
to achieve exponent 0.5.

For simplicity assume $n \in 2$

1974 Horowitz–Sahni:
Sort list of $\Sigma(J_1)$
for all $J_1 \subseteq \{1, \ldots, n/2\}$
and list of $t - \Sigma(J_2)$
for all $J_2 \subseteq \{n/2 + 1, \ldots, n$
Merge to find collisions
$\Sigma(J_1) = t - \Sigma(J_2)$,
i.e., $\Sigma(J_1 \cup J_2) = t$.

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

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 ≈ 1 after $\approx(\pi/4)2^{0.5n}$ iterations.

Left-right split (0.5)

Don't need quantum computers to achieve exponent 0.5.

For simplicity assume $n \in 2\mathbf{Z}$.

1974 Horowitz–Sahni:
Sort list of $\Sigma(J_1)$ for all $J_1 \subseteq \{1, \ldots, n/2\}$ and list of $t - \Sigma(J_2)$ for all $J_2 \subseteq \{n/2+1, \ldots, n\}$.
Merge to find collisions $\Sigma(J_1) = t - \Sigma(J_2)$, i.e., $\Sigma(J_1 \cup J_2) = t$.

is completely described
tor of two numbers
ked multiplicities):
or roots J;
or non-roots J.

+ Step 2
rly on this vector.

ompute eigenvalues
vers of this linear map
rstand evolution
of Grover's algorithm.
ability is ≈1
$\pi/4)2^{0.5n}$ iterations.

Left-right split (0.5)

Don't need quantum computers
to achieve exponent 0.5.

For simplicity assume $n \in 2\mathbf{Z}$.

1974 Horowitz–Sahni:
Sort list of $\Sigma(J_1)$
for all $J_1 \subseteq \{1, \ldots, n/2\}$
and list of $t - \Sigma(J_2)$
for all $J_2 \subseteq \{n/2+1, \ldots, n\}$.
Merge to find collisions
$\Sigma(J_1) = t - \Sigma(J_2)$,
i.e., $\Sigma(J_1 \cup J_2) = t$.

Cost $2^{0.5}$
We assig

e.g. 3663
(499, 852
4688, 59

Sort the
0, 499, 8
499 + 85
and the
36634 −
36634 −
to see th
499 + 85
36634 −

tely described
numbers
licities):
;
ts J.

s vector.

genvalues
s linear map
lution
s algorithm.
≈1
iterations.

Left-right split (0.5)

Don't need quantum computers to achieve exponent 0.5.

For simplicity assume $n \in 2\mathbf{Z}$.

1974 Horowitz–Sahni:
Sort list of $\Sigma(J_1)$ for all $J_1 \subseteq \{1, \ldots, n/2\}$ and list of $t - \Sigma(J_2)$ for all $J_2 \subseteq \{n/2+1, \ldots, n\}$.
Merge to find collisions $\Sigma(J_1) = t - \Sigma(J_2)$, i.e., $\Sigma(J_1 \cup J_2) = t$.

Cost $2^{0.5n}$ for sort
We assign cost 1 t

e.g. 36634 as sum
(499, 852, 1927, 25
4688, 5989, 6385, 7

Sort the 64 sums
0, 499, 852, 499 +
499 + 852 + 1927
and the 64 differe
36634 − 0, 36634
36634 − 4688 − ·
to see that
499 + 852 + 2535
36634 − 5989 − 638

bed

p

m.

.

Left-right split (0.5)

Don't need quantum computers to achieve exponent 0.5.

For simplicity assume $n \in 2\mathbf{Z}$.

1974 Horowitz–Sahni:
Sort list of $\Sigma(J_1)$
for all $J_1 \subseteq \{1, \ldots, n/2\}$
and list of $t - \Sigma(J_2)$
for all $J_2 \subseteq \{n/2 + 1, \ldots, n\}$.
Merge to find collisions
$\Sigma(J_1) = t - \Sigma(J_2)$,
i.e., $\Sigma(J_1 \cup J_2) = t$.

Cost $2^{0.5n}$ for sorting, mergi
We assign cost 1 to RAM.

e.g. 36634 as sum of
(499, 852, 1927, 2535, 3596, 3
4688, 5989, 6385, 7353, 7650

Sort the 64 sums
0, 499, 852, 499 + 852, . . . ,
499 + 852 + 1927 + · · · + 36
and the 64 differences
36634 − 0, 36634 − 4688, . .
36634 − 4688 − · · · − 9413
to see that
499 + 852 + 2535 + 3608 =
36634 − 5989 − 6385 − 7353 −

Left-right split (0.5)

Don't need quantum computers to achieve exponent 0.5.

For simplicity assume $n \in 2\mathbf{Z}$.

1974 Horowitz–Sahni:
Sort list of $\Sigma(J_1)$
for all $J_1 \subseteq \{1, \ldots, n/2\}$
and list of $t - \Sigma(J_2)$
for all $J_2 \subseteq \{n/2+1, \ldots, n\}$.
Merge to find collisions
$\Sigma(J_1) = t - \Sigma(J_2)$,
i.e., $\Sigma(J_1 \cup J_2) = t$.

Cost $2^{0.5n}$ for sorting, merging.
We assign cost 1 to RAM.

e.g. 36634 as sum of
$(499, 852, 1927, 2535, 3596, 3608, 4688, 5989, 6385, 7353, 7650, 9413)$:

Sort the 64 sums
$0, 499, 852, 499 + 852, \ldots,$
$499 + 852 + 1927 + \cdots + 3608$
and the 64 differences
$36634 - 0, 36634 - 4688, \ldots,$
$36634 - 4688 - \cdots - 9413$
to see that
$499 + 852 + 2535 + 3608 =$
$36634-5989-6385-7353-9413.$

nt split (0.5)

eed quantum computers
ve exponent 0.5.

plicity assume $n \in 2\mathbf{Z}$.

orowitz–Sahni:
of $\Sigma(J_1)$
$_1 \subseteq \{1, \ldots, n/2\}$
of $t - \Sigma(J_2)$
$_2 \subseteq \{n/2+1, \ldots, n\}$.
o find collisions
$t - \Sigma(J_2)$,
$J_1 \cup J_2) = t$.

Cost $2^{0.5n}$ for sorting, merging.
We assign cost 1 to RAM.

e.g. 36634 as sum of
$(499, 852, 1927, 2535, 3596, 3608,$
$4688, 5989, 6385, 7353, 7650, 9413)$:

Sort the 64 sums
$0, 499, 852, 499 + 852, \ldots,$
$499 + 852 + 1927 + \cdots + 3608$
and the 64 differences
$36634 - 0, 36634 - 4688, \ldots,$
$36634 - 4688 - \cdots - 9413$
to see that
$499 + 852 + 2535 + 3608 =$
$36634 - 5989 - 6385 - 7353 - 9413$.

Moduli (

For simp

Choose
Choose
Define t

Find all
such tha
How? S

Find all
such tha

Sort and
collisions
i.e., $\Sigma(J$

5)

um computers
nt 0.5.

ume $n \in 2\mathbf{Z}$.

hni:

$, n/2\}$
$J_2)$
$+ 1, \ldots, n\}$.
isions
),
t.

Cost $2^{0.5n}$ for sorting, merging.
We assign cost 1 to RAM.

e.g. 36634 as sum of
$(499, 852, 1927, 2535, 3596, 3608,$
$4688, 5989, 6385, 7353, 7650, 9413)$:

Sort the 64 sums
$0, 499, 852, 499 + 852, \ldots,$
$499 + 852 + 1927 + \cdots + 3608$
and the 64 differences
$36634 - 0, 36634 - 4688, \ldots,$
$36634 - 4688 - \cdots - 9413$
to see that
$499 + 852 + 2535 + 3608 =$
$36634 - 5989 - 6385 - 7353 - 9413$.

Moduli (0.5)

For simplicity assu

Choose $M \approx 2^{0.25}$
Choose $t_1 \in \{0, 1,$
Define $t_2 = t - t_1$

Find all $J_1 \subseteq \{1, .$
such that $\Sigma(J_1) \equiv$
How? Split J_1 as

Find all $J_2 \subseteq \{n/$
such that $\Sigma(J_2) \equiv$

Sort and merge to
collisions $\Sigma(J_1) =$
i.e., $\Sigma(J_1 \cup J_2) =$

ters

Z.

}.

Cost $2^{0.5n}$ for sorting, merging.
We assign cost 1 to RAM.

e.g. 36634 as sum of
$(499, 852, 1927, 2535, 3596, 3608, 4688, 5989, 6385, 7353, 7650, 9413)$:

Sort the 64 sums
$0, 499, 852, 499 + 852, \ldots,$
$499 + 852 + 1927 + \cdots + 3608$
and the 64 differences
$36634 - 0, 36634 - 4688, \ldots,$
$36634 - 4688 - \cdots - 9413$
to see that
$499 + 852 + 2535 + 3608 =$
$36634 - 5989 - 6385 - 7353 - 9413$.

Moduli (0.5)

For simplicity assume $n \in 4$

Choose $M \approx 2^{0.25n}$.
Choose $t_1 \in \{0, 1, \ldots, M -$
Define $t_2 = t - t_1$.

Find all $J_1 \subseteq \{1, \ldots, n/2\}$
such that $\Sigma(J_1) \equiv t_1 \quad$ (mo
How? Split J_1 as $J_{11} \cup J_{12}$.

Find all $J_2 \subseteq \{n/2 + 1, \ldots,$
such that $\Sigma(J_2) \equiv t_2 \quad$ (mo

Sort and merge to find all
collisions $\Sigma(J_1) = t - \Sigma(J_2)$
i.e., $\Sigma(J_1 \cup J_2) = t$.

Cost $2^{0.5n}$ for sorting, merging. We assign cost 1 to RAM.

e.g. 36634 as sum of $(499, 852, 1927, 2535, 3596, 3608, 4688, 5989, 6385, 7353, 7650, 9413)$:

Sort the 64 sums
$0, 499, 852, 499 + 852, \ldots,$
$499 + 852 + 1927 + \cdots + 3608$
and the 64 differences
$36634 - 0, 36634 - 4688, \ldots,$
$36634 - 4688 - \cdots - 9413$
to see that
$499 + 852 + 2535 + 3608 =$
$36634 - 5989 - 6385 - 7353 - 9413.$

Moduli (0.5)

For simplicity assume $n \in 4\mathbf{Z}$.

Choose $M \approx 2^{0.25n}$.
Choose $t_1 \in \{0, 1, \ldots, M - 1\}$.
Define $t_2 = t - t_1$.

Find all $J_1 \subseteq \{1, \ldots, n/2\}$
such that $\Sigma(J_1) \equiv t_1 \pmod{M}$.
How? Split J_1 as $J_{11} \cup J_{12}$.

Find all $J_2 \subseteq \{n/2 + 1, \ldots, n\}$
such that $\Sigma(J_2) \equiv t_2 \pmod{M}$.

Sort and merge to find all
collisions $\Sigma(J_1) = t - \Sigma(J_2)$,
i.e., $\Sigma(J_1 \cup J_2) = t$.

^{5n} for sorting, merging.
gn cost 1 to RAM.

34 as sum of
2, 1927, 2535, 3596, 3608,
89, 6385, 7353, 7650, 9413):

64 sums
52, 499 + 852, . . . ,
52 + 1927 + · · · + 3608
64 differences
0, 36634 − 4688, . . . ,
4688 − · · · − 9413
hat
52 + 2535 + 3608 =
5989 − 6385 − 7353 − 9413.

Moduli (0.5)

For simplicity assume $n \in 4\mathbf{Z}$.

Choose $M \approx 2^{0.25n}$.
Choose $t_1 \in \{0, 1, \ldots, M - 1\}$.
Define $t_2 = t - t_1$.

Find all $J_1 \subseteq \{1, \ldots, n/2\}$
such that $\Sigma(J_1) \equiv t_1 \pmod{M}$.
How? Split J_1 as $J_{11} \cup J_{12}$.

Find all $J_2 \subseteq \{n/2 + 1, \ldots, n\}$
such that $\Sigma(J_2) \equiv t_2 \pmod{M}$.

Sort and merge to find all
collisions $\Sigma(J_1) = t - \Sigma(J_2)$,
i.e., $\Sigma(J_1 \cup J_2) = t$.

Finds J
There ar
Each ch
Total co

Not visib
this uses
assuming

Algorith
introduc
2006 Els
2010 Ho

Different
for simil
1981 Sc

ting, merging.
to RAM.

of
535, 3596, 3608,
7353, 7650, 9413):

852, . . . ,
$+ \cdots + 3608$
nces
$- 4688, \ldots,$
$\cdot - 9413$

$+ 3608 =$
$35 - 7353 - 9413.$

Moduli (0.5)

For simplicity assume $n \in 4\mathbf{Z}$.

Choose $M \approx 2^{0.25n}$.
Choose $t_1 \in \{0, 1, \ldots, M-1\}$.
Define $t_2 = t - t_1$.

Find all $J_1 \subseteq \{1, \ldots, n/2\}$
such that $\Sigma(J_1) \equiv t_1 \pmod{M}$.
How? Split J_1 as $J_{11} \cup J_{12}$.

Find all $J_2 \subseteq \{n/2 + 1, \ldots, n\}$
such that $\Sigma(J_2) \equiv t_2 \pmod{M}$.

Sort and merge to find all
collisions $\Sigma(J_1) = t - \Sigma(J_2)$,
i.e., $\Sigma(J_1 \cup J_2) = t$.

Finds J iff $\Sigma(J_1) \equiv$
There are $\approx 2^{0.25n}$
Each choice costs
Total cost $2^{0.5n}$.

Not visible in cost
this uses space on
assuming typical d

Algorithm has bee
introduced at leas
2006 Elsenhans–Ja
2010 Howgrave-Gr

Different techniqu
for similar space re
1981 Schroeppel–S

ing.

3608,

, 9413):

608

. . ,

−9413.

Moduli (0.5)

For simplicity assume $n \in 4\mathbf{Z}$.

Choose $M \approx 2^{0.25n}$.
Choose $t_1 \in \{0, 1, \ldots, M-1\}$.
Define $t_2 = t - t_1$.

Find all $J_1 \subseteq \{1, \ldots, n/2\}$
such that $\Sigma(J_1) \equiv t_1 \pmod{M}$.
How? Split J_1 as $J_{11} \cup J_{12}$.

Find all $J_2 \subseteq \{n/2+1, \ldots, n\}$
such that $\Sigma(J_2) \equiv t_2 \pmod{M}$.

Sort and merge to find all
collisions $\Sigma(J_1) = t - \Sigma(J_2)$,
i.e., $\Sigma(J_1 \cup J_2) = t$.

Finds J iff $\Sigma(J_1) \equiv t_1$.
There are $\approx 2^{0.25n}$ choices o
Each choice costs $2^{0.25n}$.
Total cost $2^{0.5n}$.

Not visible in cost metric:
this uses space only $2^{0.25n}$,
assuming typical distributior

Algorithm has been
introduced at least twice:
2006 Elsenhans–Jahnel;
2010 Howgrave-Graham–Jou

Different technique
for similar space reduction:
1981 Schroeppel–Shamir.

Moduli (0.5)

For simplicity assume $n \in 4\mathbf{Z}$.

Choose $M \approx 2^{0.25n}$.
Choose $t_1 \in \{0, 1, \ldots, M-1\}$.
Define $t_2 = t - t_1$.

Find all $J_1 \subseteq \{1, \ldots, n/2\}$
such that $\Sigma(J_1) \equiv t_1 \pmod{M}$.
How? Split J_1 as $J_{11} \cup J_{12}$.

Find all $J_2 \subseteq \{n/2+1, \ldots, n\}$
such that $\Sigma(J_2) \equiv t_2 \pmod{M}$.

Sort and merge to find all
collisions $\Sigma(J_1) = t - \Sigma(J_2)$,
i.e., $\Sigma(J_1 \cup J_2) = t$.

Finds J iff $\Sigma(J_1) \equiv t_1$.
There are $\approx 2^{0.25n}$ choices of t_1.
Each choice costs $2^{0.25n}$.
Total cost $2^{0.5n}$.

Not visible in cost metric:
this uses space only $2^{0.25n}$,
assuming typical distribution.

Algorithm has been
introduced at least twice:
2006 Elsenhans–Jahnel;
2010 Howgrave-Graham–Joux.

Different technique
for similar space reduction:
1981 Schroeppel–Shamir.

(0.5)

plicity assume $n \in 4\mathbf{Z}$.

$M \approx 2^{0.25n}$.

$t_1 \in \{0, 1, \ldots, M-1\}$.

$_2 = t - t_1$.

$J_1 \subseteq \{1, \ldots, n/2\}$

nt $\Sigma(J_1) \equiv t_1 \pmod{M}$.

plit J_1 as $J_{11} \cup J_{12}$.

$J_2 \subseteq \{n/2+1, \ldots, n\}$

nt $\Sigma(J_2) \equiv t_2 \pmod{M}$.

l merge to find all

s $\Sigma(J_1) = t - \Sigma(J_2)$,

$J_1 \cup J_2) = t$.

Finds J iff $\Sigma(J_1) \equiv t_1$.

There are $\approx 2^{0.25n}$ choices of t_1.

Each choice costs $2^{0.25n}$.

Total cost $2^{0.5n}$.

Not visible in cost metric:

this uses space only $2^{0.25n}$,

assuming typical distribution.

Algorithm has been

introduced at least twice:

2006 Elsenhans–Jahnel;

2010 Howgrave-Graham–Joux.

Different technique

for similar space reduction:

1981 Schroeppel–Shamir.

e.g. $M =$

(499, 85

4688, 59

Try each

In partic

There ar

(499, 85

with sum

There ar

(4688, 5

with sum

Sort and

$499 + 8$

$36634 -$

ıme $n \in 4\mathbf{Z}$.

n.

$\ldots, M-1\}$.

.

$.., n/2\}$

$\equiv t_1 \pmod{M}$.

$J_{11} \cup J_{12}$.

$2+1, \ldots, n\}$

$\equiv t_2 \pmod{M}$.

find all

$t - \Sigma(J_2)$,

t.

Finds J iff $\Sigma(J_1) \equiv t_1$.
There are $\approx 2^{0.25n}$ choices of t_1.
Each choice costs $2^{0.25n}$.
Total cost $2^{0.5n}$.

Not visible in cost metric:
this uses space only $2^{0.25n}$,
assuming typical distribution.

Algorithm has been
introduced at least twice:
2006 Elsenhans–Jahnel;
2010 Howgrave-Graham–Joux.

Different technique
for similar space reduction:
1981 Schroeppel–Shamir.

e.g. $M = 8$, $t = 3$
(499, 852, 1927, 25
4688, 5989, 6385, 7

Try each $t_1 \in \{0,$

In particular try t_1
There are 12 subse
(499, 852, 1927, 25
with sum 6 modul
There are 6 subse
(4688, 5989, 6385,
with sum 36634 –
Sort and merge to
499 + 852 + 2535
36634 − 5989 − 638

Z.

1}.

d M).

$n\}$
d M).

),

Finds J iff $\Sigma(J_1) \equiv t_1$.
There are $\approx 2^{0.25n}$ choices of t_1.
Each choice costs $2^{0.25n}$.
Total cost $2^{0.5n}$.

Not visible in cost metric:
this uses space only $2^{0.25n}$,
assuming typical distribution.

Algorithm has been
introduced at least twice:
2006 Elsenhans–Jahnel;
2010 Howgrave-Graham–Joux.

Different technique
for similar space reduction:
1981 Schroeppel–Shamir.

e.g. $M = 8$, $t = 36634$, $x =$
$(499, 852, 1927, 2535, 3596,$
$4688, 5989, 6385, 7353, 7650$

Try each $t_1 \in \{0, 1, \ldots, 7\}$.

In particular try $t_1 = 6$.
There are 12 subsequences o
$(499, 852, 1927, 2535, 3596,$
with sum 6 modulo 8.
There are 6 subsequences of
$(4688, 5989, 6385, 7353, 765$
with sum $36634 - 6$ modulo
Sort and merge to find
$499 + 852 + 2535 + 3608 =$
$36634 - 5989 - 6385 - 7353 -$

Finds J iff $\Sigma(J_1) \equiv t_1$.
There are $\approx 2^{0.25n}$ choices of t_1.
Each choice costs $2^{0.25n}$.
Total cost $2^{0.5n}$.

Not visible in cost metric:
this uses space only $2^{0.25n}$,
assuming typical distribution.

Algorithm has been
introduced at least twice:
2006 Elsenhans–Jahnel;
2010 Howgrave-Graham–Joux.

Different technique
for similar space reduction:
1981 Schroeppel–Shamir.

e.g. $M = 8$, $t = 36634$, $x =$
$(499, 852, 1927, 2535, 3596, 3608,$
$4688, 5989, 6385, 7353, 7650, 9413)$:

Try each $t_1 \in \{0, 1, \ldots, 7\}$.

In particular try $t_1 = 6$.
There are 12 subsequences of
$(499, 852, 1927, 2535, 3596, 3608)$
with sum 6 modulo 8.
There are 6 subsequences of
$(4688, 5989, 6385, 7353, 7650, 9413)$
with sum $36634 - 6$ modulo 8.
Sort and merge to find
$499 + 852 + 2535 + 3608 =$
$36634 - 5989 - 6385 - 7353 - 9413$.

iff $\Sigma(J_1) \equiv t_1$.
re $\approx 2^{0.25n}$ choices of t_1.
oice costs $2^{0.25n}$.
st $2^{0.5n}$.

ble in cost metric:
s space only $2^{0.25n}$,
g typical distribution.

m has been
ed at least twice:
senhans–Jahnel;
owgrave-Graham–Joux.

t technique
ar space reduction:
hroeppel–Shamir.

e.g. $M = 8$, $t = 36634$, $x =$
$(499, 852, 1927, 2535, 3596, 3608,$
$4688, 5989, 6385, 7353, 7650, 9413)$:

Try each $t_1 \in \{0, 1, \ldots, 7\}$.

In particular try $t_1 = 6$.
There are 12 subsequences of
$(499, 852, 1927, 2535, 3596, 3608)$
with sum 6 modulo 8.
There are 6 subsequences of
$(4688, 5989, 6385, 7353, 7650, 9413)$
with sum $36634 - 6$ modulo 8.
Sort and merge to find
$499 + 852 + 2535 + 3608 =$
$36634 - 5989 - 6385 - 7353 - 9413$.

Quantur

Cost $2^{n/}$
1998 Bra

For simp

Comput
$J_1 \subseteq \{1,$
Sort $L =$

Can now
$J_2 \mapsto [t$
for $J_2 \subseteq$
Recall: v

Use Grov
whether

$\equiv t_1$.
choices of t_1.
$2^{0.25n}$.

metric:
ly $2^{0.25n}$,
listribution.

n
t twice:
ahnel;
raham–Joux.

e
eduction:
Shamir.

e.g. $M = 8$, $t = 36634$, $x =$ $(499, 852, 1927, 2535, 3596, 3608, 4688, 5989, 6385, 7353, 7650, 9413)$:

Try each $t_1 \in \{0, 1, \ldots, 7\}$.

In particular try $t_1 = 6$.
There are 12 subsequences of $(499, 852, 1927, 2535, 3596, 3608)$ with sum 6 modulo 8.
There are 6 subsequences of $(4688, 5989, 6385, 7353, 7650, 9413)$ with sum $36634 - 6$ modulo 8.
Sort and merge to find
$499 + 852 + 2535 + 3608 =$
$36634 - 5989 - 6385 - 7353 - 9413$.

Quantum left-righ

Cost $2^{n/3}$, imitati
1998 Brassard–Hø

For simplicity assu

Compute $\Sigma(J_1)$ fo
$J_1 \subseteq \{1, 2, \ldots, n/$
Sort $L = \{\Sigma(J_1)\}$.

Can now efficiently
$J_2 \mapsto [t - \Sigma(J_2) \notin$
for $J_2 \subseteq \{n/3 + 1$
Recall: we assign

Use Grover's meth
whether this funct

f t_1.

n.

ux.

e.g. $M = 8$, $t = 36634$, $x =$
$(499, 852, 1927, 2535, 3596, 3608,$
$4688, 5989, 6385, 7353, 7650, 9413)$:

Try each $t_1 \in \{0, 1, \ldots, 7\}$.

In particular try $t_1 = 6$.
There are 12 subsequences of
$(499, 852, 1927, 2535, 3596, 3608)$
with sum 6 modulo 8.
There are 6 subsequences of
$(4688, 5989, 6385, 7353, 7650, 9413)$
with sum $36634 - 6$ modulo 8.
Sort and merge to find
$499 + 852 + 2535 + 3608 =$
$36634 - 5989 - 6385 - 7353 - 9413$.

Quantum left-right split (0.3

Cost $2^{n/3}$, imitating
1998 Brassard–Høyer–Tapp:

For simplicity assume $n \in 3$

Compute $\Sigma(J_1)$ for all
$J_1 \subseteq \{1, 2, \ldots, n/3\}$.
Sort $L = \{\Sigma(J_1)\}$.

Can now efficiently compute
$J_2 \mapsto [t - \Sigma(J_2) \notin L]$
for $J_2 \subseteq \{n/3 + 1, \ldots, n\}$.
Recall: we assign cost 1 to

Use Grover's method to see
whether this function has a

e.g. $M = 8$, $t = 36634$, $x =$ $(499, 852, 1927, 2535, 3596, 3608, 4688, 5989, 6385, 7353, 7650, 9413)$:

Try each $t_1 \in \{0, 1, \ldots, 7\}$.

In particular try $t_1 = 6$.
There are 12 subsequences of $(499, 852, 1927, 2535, 3596, 3608)$ with sum 6 modulo 8.
There are 6 subsequences of $(4688, 5989, 6385, 7353, 7650, 9413)$ with sum $36634 - 6$ modulo 8.
Sort and merge to find $499 + 852 + 2535 + 3608 = 36634 - 5989 - 6385 - 7353 - 9413$.

<u>Quantum left-right split (0.333...)</u>

Cost $2^{n/3}$, imitating 1998 Brassard–Høyer–Tapp:

For simplicity assume $n \in 3\mathbf{Z}$.

Compute $\Sigma(J_1)$ for all $J_1 \subseteq \{1, 2, \ldots, n/3\}$.
Sort $L = \{\Sigma(J_1)\}$.

Can now efficiently compute $J_2 \mapsto [t - \Sigma(J_2) \notin L]$ for $J_2 \subseteq \{n/3 + 1, \ldots, n\}$.
Recall: we assign cost 1 to RAM.

Use Grover's method to see whether this function has a root.

$= 8$, $t = 36634$, $x =$
$2, 1927, 2535, 3596, 3608,$
$89, 6385, 7353, 7650, 9413)$:

n $t_1 \in \{0, 1, \ldots, 7\}$.

cular try $t_1 = 6$.
re 12 subsequences of
$2, 1927, 2535, 3596, 3608)$
n 6 modulo 8.
re 6 subsequences of
$989, 6385, 7353, 7650, 9413)$
n $36634 - 6$ modulo 8.
l merge to find
$52 + 2535 + 3608 =$
$5989 - 6385 - 7353 - 9413$.

Quantum left-right split (0.333 . . .)

Cost $2^{n/3}$, imitating
1998 Brassard–Høyer–Tapp:

For simplicity assume $n \in 3\mathbf{Z}$.

Compute $\Sigma(J_1)$ for all
$J_1 \subseteq \{1, 2, \ldots, n/3\}$.
Sort $L = \{\Sigma(J_1)\}$.

Can now efficiently compute
$J_2 \mapsto [t - \Sigma(J_2) \notin L]$
for $J_2 \subseteq \{n/3 + 1, \ldots, n\}$.
Recall: we assign cost 1 to RAM.

Use Grover's method to see
whether this function has a root.

Quantum

Unique-c
Say f ha
exactly c
i.e., $p \neq$
Problem

Cost 2^n
the set c
Compute

Generali
success p
Choose a
Compute

6634, $x =$
535, 3596, 3608,
7353, 7650, 9413):

1, . . . , 7}.

= 6.
equences of
535, 3596, 3608)
o 8.
quences of
7353, 7650, 9413)
6 modulo 8.
find
+ 3608 =
35 − 7353 − 9413.

Quantum left-right split (0.333 . . .)

Cost $2^{n/3}$, imitating 1998 Brassard–Høyer–Tapp:

For simplicity assume $n \in 3\mathbf{Z}$.

Compute $\Sigma(J_1)$ for all $J_1 \subseteq \{1, 2, \ldots, n/3\}$. Sort $L = \{\Sigma(J_1)\}$.

Can now efficiently compute $J_2 \mapsto [t - \Sigma(J_2) \notin L]$ for $J_2 \subseteq \{n/3 + 1, \ldots, n\}$. Recall: we assign cost 1 to RAM.

Use Grover's method to see whether this function has a root.

Quantum walk

Unique-collision-fi
Say f has n-bit in
exactly one collisio
i.e., $p \neq q$, $f(p) =$
Problem: find this

Cost 2^n: Define S
the set of n-bit st
Compute $f(S)$, so

Generalize to cost
success probability
Choose a set S of
Compute $f(S)$, so

3608,
, 9413):

of
3608)

f
0, 9413)
8.

–9413.

Quantum left-right split (0.333...)

Cost $2^{n/3}$, imitating
1998 Brassard–Høyer–Tapp:

For simplicity assume $n \in 3\mathbf{Z}$.

Compute $\Sigma(J_1)$ for all
$J_1 \subseteq \{1, 2, \ldots, n/3\}$.
Sort $L = \{\Sigma(J_1)\}$.

Can now efficiently compute
$J_2 \mapsto [t - \Sigma(J_2) \notin L]$
for $J_2 \subseteq \{n/3 + 1, \ldots, n\}$.
Recall: we assign cost 1 to RAM.

Use Grover's method to see
whether this function has a root.

Quantum walk

Unique-collision-finding prob
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 $\approx(r/2^n)$
Choose a set S of size r.
Compute $f(S)$, sort.

Quantum left-right split (0.333...)

Cost $2^{n/3}$, imitating 1998 Brassard–Høyer–Tapp:

For simplicity assume $n \in 3\mathbf{Z}$.

Compute $\Sigma(J_1)$ for all $J_1 \subseteq \{1, 2, \ldots, n/3\}$. Sort $L = \{\Sigma(J_1)\}$.

Can now efficiently compute $J_2 \mapsto [t - \Sigma(J_2) \notin L]$ for $J_2 \subseteq \{n/3 + 1, \ldots, n\}$. Recall: we assign cost 1 to RAM.

Use Grover's method to see whether this function has a root.

Quantum walk

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.

m left-right split $(0.333\ldots)$

$/3$, imitating
assard–Høyer–Tapp:

plicity assume $n \in 3\mathbf{Z}$.

e $\Sigma(J_1)$ for all
, $2, \ldots, n/3\}$.
$= \{\Sigma(J_1)\}$.

v efficiently compute
$- \Sigma(J_2) \notin L]$
$\{n/3+1, \ldots, n\}$.
we assign cost 1 to RAM.

ver's method to see
this function has a root.

Quantum walk

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 str
the gene
the set S
the num

Very effi
to $D(T)$
$\#S = \#$

2003 An
Magniez
Create s
$(D(S), L$
By a qua
find S c

t split (0.333...)

ng

yer–Tapp:

me $n \in 3\mathbf{Z}$.

r all

3}.

y compute

$\notin L$]

$, \ldots, n\}$.

cost 1 to RAM.

nod to see

ion has a root.

Quantum walk

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 $D($

the generalized co

the set S; the mul

the number of coll

Very efficient to m

to $D(T)$ if T is an

$\#S = \#T = r$, $\#$

2003 Ambainis, sir

Magniez–Nayak–R

Create superpositi

$(D(S), D(T))$ with

By a quantum wal

find S containing

333 . . .)

Z.

e

RAM.

root.

Quantum walk

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 $D(S)$ captur
the generalized computation
the set S; the multiset $f(S)$
the number of collisions in S

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

2003 Ambainis, simplified 20
Magniez–Nayak–Roland–San
Create superposition of state
$(D(S), D(T))$ with adjacent
By a quantum walk
find S containing a collision

Quantum walk

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.

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.

m walk

collision-finding problem:
as n-bit inputs,
one collision $\{p, q\}$:
q, $f(p) = f(q)$.
: find this collision.

: Define S as
of n-bit strings.
e $f(S)$, sort.

ze to cost r,
probability $\approx(r/2^n)^2$:
a set S of size r.
e $f(S)$, sort.

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

Start fro
Repeat ≈
Negat
if S
Repea
For
D
For
D

Now hig
that T c
Cost $r +$

nding problem:
puts,
on $\{p, q\}$:
$= f(q)$.
s collision.

S as
rings.
rt.

r,
$\approx (r/2^n)^2$:
size r.
rt.

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

Start from uniform
Repeat $\approx 0.6 \cdot 2^n/$
Negate $a_{S,T}$
if S contains
Repeat $\approx 0.7 \cdot \surd$
For each T:
Diffuse $a_{S,}$
For each S:
Diffuse $a_{S,}$

Now high probabil
that T contains co
Cost $r + 2^n/\sqrt{r}$.

blem:

2:

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 work

Start from uniform superpos

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 a

For each S:

Diffuse $a_{S,T}$ across a

Now high probability that T contains collision. Cost $r + 2^n/\sqrt{r}$. Optimize:

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: $2^{2n/3}$.

ucture $D(S)$ capturing
eralized computation:
S; the multiset $f(S)$;
ber of collisions in S.

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

nbainis, simplified 2007
–Nayak–Roland–Santha:
uperposition of states
$D(T)$) with adjacent S, T.
antum walk
ontaining 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
$(\#(S \cap$
reduce a
Analyze

e.g. $n =$
0 negati

Pr[class
Pr[class
Pr[class
Pr[class
Pr[class
Pr[class
Pr[class

Right co

S) capturing
mputation:
ltiset $f(S)$;
lisions in S.

nove from $D(S)$
n **adjacent** set:
$(S \cap T) = r - 1$.

mplified 2007
Roland–Santha:
on of states
n adjacent S, T.
lk
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) acc
$(\#(S \cap \{p, q\}), \#$
reduce a to low-di
Analyze evolution

e.g. $n = 15$, $r = 1$
0 negations and 0

Pr[class $(0, 0)$] $\approx$ 0
Pr[class $(0, 1)$] $\approx$ 0
Pr[class $(1, 0)$] $\approx$ 0
Pr[class $(1, 1)$] $\approx$ 0
Pr[class $(1, 2)$] $\approx$ 0
Pr[class $(2, 1)$] $\approx$ 0
Pr[class $(2, 2)$] $\approx$ 0

Right column is si

ing

:

);

S.

$D(S)$

t set:

$= r - 1$.

007

ntha:

es

. S, T.

.

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 vec

e.g. $n = 15$, $r = 1024$, after

0 negations and 0 diffusions

Pr[class $(0, 0)$] $\approx 0.938; +$

Pr[class $(0, 1)$] $\approx 0.000; +$

Pr[class $(1, 0)$] $\approx 0.000; +$

Pr[class $(1, 1)$] $\approx 0.060; +$

Pr[class $(1, 2)$] $\approx 0.000; +$

Pr[class $(2, 1)$] $\approx 0.000; +$

Pr[class $(2, 2)$] $\approx 0.001; +$

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

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

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

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 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 $a_{S,T}$.

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

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 $a_{S,T}$.

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
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 $a_{S,T}$.

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 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 $a_{S,T}$.

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 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 $a_{S,T}$.

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
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 $a_{S,T}$.

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 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 $a_{S,T}$.

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 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 $a_{S,T}$.

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
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 $a_{S,T}$.

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
10 negations and 460 diffusions:

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

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

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
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 $a_{S,T}$.

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 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 $a_{S,T}$.

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
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 $a_{S,T}$.

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
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 $a_{S,T}$.

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

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 $a_{S,T}$.

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 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 $a_{S,T}$.

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
17 negations and 782 diffusions:

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

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

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
18 negations and 828 diffusions:

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

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

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
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 $a_{S,T}$.

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

20 negations and 920 diffusions:

Pr[class $(0, 0)$] $\approx 0.093; +$

Pr[class $(0, 1)$] $\approx 0.007; +$

Pr[class $(1, 0)$] $\approx 0.007; -$

Pr[class $(1, 1)$] $\approx 0.007; +$

Pr[class $(1, 2)$] $\approx 0.027; +$

Pr[class $(2, 1)$] $\approx 0.002; +$

Pr[class $(2, 2)$] $\approx 0.857; +$

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

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
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 $a_{S,T}$.

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
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 $a_{S,T}$.

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
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 $a_{S,T}$.

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
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 $a_{S,T}$.

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

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 $a_{S,T}$.

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

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 $a_{S,T}$.

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 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 $a_{S,T}$.

e quantum walk works:

om uniform superposition.

≈$0.6 \cdot 2^n/r$ times:

ze $a_{S,T}$

T contains collision.

nt ≈$0.7 \cdot \sqrt{r}$ times:

each T:

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

each S:

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

h probability

contains collision.

$2^n/\sqrt{r}$. 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

27 negations and 1242 diffusions:

Pr[class $(0, 0)$] $\approx 0.011; -$

Pr[class $(0, 1)$] $\approx 0.007; +$

Pr[class $(1, 0)$] $\approx 0.007; -$

Pr[class $(1, 1)$] $\approx 0.001; -$

Pr[class $(1, 2)$] $\approx 0.034; +$

Pr[class $(2, 1)$] $\approx 0.003; +$

Pr[class $(2, 2)$] $\approx 0.938; +$

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

Subset-s

Conside

$f(1, J_1)$

for $J_1 \subseteq$

$f(2, J_2)$

for $J_2 \subseteq$

Good ch

collision

$n/2 + 1$

so quant

Easily tw

to handl

ignore Σ

walk works:

n superposition.

r times:

collision.

$\sqrt{r}$ times:

$_T$ across all S.

$_T$ across all T.

ity

ollision.

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 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 $a_{S,T}$.

Subset-sum walk (

Consider f defined
$f(1, J_1) = \Sigma(J_1)$
for $J_1 \subseteq \{1, \ldots, n$
$f(2, J_2) = t - \Sigma($
for $J_2 \subseteq \{n/2 + 1$

Good chance of u
collision $\Sigma(J_1) = t$

$n/2 + 1$ bits of inp
so quantum walk

Easily tweak quan
to handle more co
ignore $\Sigma(J_1) = \Sigma($

ks:

sition.

ll S.

ll T.

$2^{2n/3}$.

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 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 $a_{S,T}$.

Subset-sum walk (0.333 . . .)

Consider f defined by
$f(1, J_1) = \Sigma(J_1)$
for $J_1 \subseteq \{1, \ldots, n/2\}$;
$f(2, J_2) = t - \Sigma(J_2)$
for $J_2 \subseteq \{n/2 + 1, \ldots, n\}$.

Good chance of unique collision $\Sigma(J_1) = t - \Sigma(J_2)$.

$n/2 + 1$ bits of input, so quantum walk costs $2^{n/3}$

Easily tweak quantum walk to handle more collisions, ignore $\Sigma(J_1) = \Sigma(J_1')$, etc.

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 $a_{S,T}$.

Subset-sum walk (0.333...)

Consider f defined by
$f(1, J_1) = \Sigma(J_1)$
for $J_1 \subseteq \{1, \ldots, n/2\}$;
$f(2, J_2) = t - \Sigma(J_2)$
for $J_2 \subseteq \{n/2 + 1, \ldots, n\}$.

Good chance of unique collision $\Sigma(J_1) = t - \Sigma(J_2)$.

$n/2 + 1$ bits of input, so quantum walk costs $2^{n/3}$.

Easily tweak quantum walk to handle more collisions, ignore $\Sigma(J_1) = \Sigma(J_1')$, etc.

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

$= 15$, $r = 1024$, after
tions and 1242 diffusions:

$(0, 0)] \approx 0.011; -$
$(0, 1)] \approx 0.007; +$
$(1, 0)] \approx 0.007; -$
$(1, 1)] \approx 0.001; -$
$(1, 2)] \approx 0.034; +$
$(2, 1)] \approx 0.003; +$
$(2, 2)] \approx 0.938; +$

olumn is sign of $a_{S,T}$.

Subset-sum walk (0.333...)

Consider f defined by
$f(1, J_1) = \Sigma(J_1)$
for $J_1 \subseteq \{1, \ldots, n/2\}$;
$f(2, J_2) = t - \Sigma(J_2)$
for $J_2 \subseteq \{n/2 + 1, \ldots, n\}$.

Good chance of unique
collision $\Sigma(J_1) = t - \Sigma(J_2)$.

$n/2 + 1$ bits of input,
so quantum walk costs $2^{n/3}$.

Easily tweak quantum walk
to handle more collisions,
ignore $\Sigma(J_1) = \Sigma(J_1')$, etc.

Generali

Choose
(Origina
is the sp

Take set
$J_{11} \in S_1$
(Origina
of *all* J_1
Compute
for each

Similarly
subsets
Compute
for each

cording to
$(T \cap \{p, q\}))$;
m vector.
of this vector.

.024, after
1242 diffusions:

0.011; $-$
0.007; $+$
0.007; $-$
0.001; $-$
0.034; $+$
0.003; $+$
0.938; $+$

gn of $a_{S,T}$.

Subset-sum walk (0.333...)

Consider f defined by
$f(1, J_1) = \Sigma(J_1)$
for $J_1 \subseteq \{1, \ldots, n/2\}$;
$f(2, J_2) = t - \Sigma(J_2)$
for $J_2 \subseteq \{n/2 + 1, \ldots, n\}$.

Good chance of unique
collision $\Sigma(J_1) = t - \Sigma(J_2)$.

$n/2 + 1$ bits of input,
so quantum walk costs $2^{n/3}$.

Easily tweak quantum walk
to handle more collisions,
ignore $\Sigma(J_1) = \Sigma(J_1')$, etc.

Generalized modul

Choose M, t_1, r w
(Original moduli a
is the special case

Take set S_{11}, $\#S_1$
$J_{11} \in S_{11} \Rightarrow J_{11} \subseteq$
(Original algorithn
of *all* $J_{11} \subseteq \{1, \ldots$
Compute $\Sigma(J_{11})$ n
for each $J_{11} \in S_{11}$

Similarly take a se
subsets of $\{n/4 +$
Compute $t_1 - \Sigma($
for each $J_{12} \in S_{12}$

}));

ctor.

r

sions:

r.

Subset-sum walk (0.333 . . .)

Consider f defined by
$f(1, J_1) = \Sigma(J_1)$
for $J_1 \subseteq \{1, \ldots, n/2\}$;
$f(2, J_2) = t - \Sigma(J_2)$
for $J_2 \subseteq \{n/2 + 1, \ldots, n\}$.

Good chance of unique collision $\Sigma(J_1) = t - \Sigma(J_2)$.

$n/2 + 1$ bits of input, so quantum walk costs $2^{n/3}$.

Easily tweak quantum walk to handle more collisions, ignore $\Sigma(J_1) = \Sigma(J_1')$, etc.

Generalized moduli

Choose M, t_1, r with $M \approx$
(Original moduli algorithm
is the special case $r = 2^{n/4}$.

Take set S_{11}, $\#S_{11} = r$, wh
$J_{11} \in S_{11} \Rightarrow J_{11} \subseteq \{1, \ldots,$
(Original algorithm: S_{11} is t
of *all* $J_{11} \subseteq \{1, \ldots, n/4\}$.)
Compute $\Sigma(J_{11})$ mod M
for each $J_{11} \in S_{11}$.

Similarly take a set S_{12} of r
subsets of $\{n/4 + 1, \ldots, n/$
Compute $t_1 - \Sigma(J_{12})$ mod
for each $J_{12} \in S_{12}$.

Subset-sum walk (0.333...)

Consider f defined by
$f(1, J_1) = \Sigma(J_1)$
for $J_1 \subseteq \{1, \ldots, n/2\}$;
$f(2, J_2) = t - \Sigma(J_2)$
for $J_2 \subseteq \{n/2+1, \ldots, n\}$.

Good chance of unique collision $\Sigma(J_1) = t - \Sigma(J_2)$.

$n/2+1$ bits of input, so quantum walk costs $2^{n/3}$.

Easily tweak quantum walk to handle more collisions, ignore $\Sigma(J_1) = \Sigma(J_1')$, etc.

Generalized moduli

Choose M, t_1, r with $M \approx r$.
(Original moduli algorithm is the special case $r = 2^{n/4}$.)

Take set S_{11}, $\#S_{11} = r$, where $J_{11} \in S_{11} \Rightarrow J_{11} \subseteq \{1, \ldots, n/4\}$.
(Original algorithm: S_{11} is the set of *all* $J_{11} \subseteq \{1, \ldots, n/4\}$.)
Compute $\Sigma(J_{11}) \bmod M$ for each $J_{11} \in S_{11}$.

Similarly take a set S_{12} of r subsets of $\{n/4+1, \ldots, n/2\}$.
Compute $t_1 - \Sigma(J_{12}) \bmod M$ for each $J_{12} \in S_{12}$.

sum walk (0.333 . . .)

r f defined by

$= \Sigma(J_1)$

$\{1, \ldots, n/2\}$;

$= t - \Sigma(J_2)$

$\{n/2 + 1, \ldots, n\}$.

hance of unique

$\Sigma(J_1) = t - \Sigma(J_2)$.

bits of input,

tum walk costs $2^{n/3}$.

weak quantum walk

e more collisions,

$\Sigma(J_1) = \Sigma(J_1')$, etc.

Generalized moduli

Choose M, t_1, r with $M \approx r$.
(Original moduli algorithm
is the special case $r = 2^{n/4}$.)

Take set S_{11}, $\#S_{11} = r$, where
$J_{11} \in S_{11} \Rightarrow J_{11} \subseteq \{1, \ldots, n/4\}$.
(Original algorithm: S_{11} is the set
of *all* $J_{11} \subseteq \{1, \ldots, n/4\}$.)
Compute $\Sigma(J_{11})$ mod M
for each $J_{11} \in S_{11}$.

Similarly take a set S_{12} of r
subsets of $\{n/4 + 1, \ldots, n/2\}$.
Compute $t_1 - \Sigma(J_{12})$ mod M
for each $J_{12} \in S_{12}$.

Find all
$\Sigma(J_{11}) \equiv$
i.e., $\Sigma(J$
where J_1
Compute

Similarly
list of J_2
$\Rightarrow$ each

Find col

Success
at findin
$\Sigma(J) =$

Assumin
cost r, s

(0.333 . . .)

d by

$/2\}$;

$J_2)$

$, \ldots, n\}$.

nique

$t - \Sigma(J_2)$.

put,

costs $2^{n/3}$.

tum walk

llisions,

(J'_1), etc.

Generalized moduli

Choose M, t_1, r with $M \approx r$.
(Original moduli algorithm
is the special case $r = 2^{n/4}$.)

Take set S_{11}, $\#S_{11} = r$, where
$J_{11} \in S_{11} \Rightarrow J_{11} \subseteq \{1, \ldots, n/4\}$.
(Original algorithm: S_{11} is the set
of *all* $J_{11} \subseteq \{1, \ldots, n/4\}$.)
Compute $\Sigma(J_{11})$ mod M
for each $J_{11} \in S_{11}$.

Similarly take a set S_{12} of r
subsets of $\{n/4 + 1, \ldots, n/2\}$.
Compute $t_1 - \Sigma(J_{12})$ mod M
for each $J_{12} \in S_{12}$.

Find all collisions

$\Sigma(J_{11}) \equiv t_1 - \Sigma($

i.e., $\Sigma(J_1) \equiv t_1$ (

where $J_1 = J_{11} \cup$

Compute each $\Sigma($

Similarly S_{21}, S_{22}

list of J_2 with $\Sigma($

$\Rightarrow$ each $t - \Sigma(J_2)$

Find collisions $\Sigma($

Success probability

at finding any part

$\Sigma(J) = t$, $\Sigma(J_1) \equiv$

Assuming typical

cost r, since $M \approx$

Generalized moduli

Choose M, t_1, r with $M \approx r$.
(Original moduli algorithm is the special case $r = 2^{n/4}$.)

Take set S_{11}, $\#S_{11} = r$, where $J_{11} \in S_{11} \Rightarrow J_{11} \subseteq \{1, \ldots, n/4\}$.
(Original algorithm: S_{11} is the set of *all* $J_{11} \subseteq \{1, \ldots, n/4\}$.)
Compute $\Sigma(J_{11}) \bmod M$ for each $J_{11} \in S_{11}$.

Similarly take a set S_{12} of r subsets of $\{n/4+1, \ldots, n/2\}$.
Compute $t_1 - \Sigma(J_{12}) \bmod M$ for each $J_{12} \in S_{12}$.

Find all collisions
$\Sigma(J_{11}) \equiv t_1 - \Sigma(J_{12})$,
i.e., $\Sigma(J_1) \equiv t_1 \pmod{M}$
where $J_1 = J_{11} \cup J_{12}$.
Compute each $\Sigma(J_1)$.

Similarly S_{21}, $S_{22} \Rightarrow$
list of J_2 with $\Sigma(J_2) \equiv t - t$
$\Rightarrow$ each $t - \Sigma(J_2)$.

Find collisions $\Sigma(J_1) = t -$

Success probability $r^4/2^n$
at finding any particular J w
$\Sigma(J) = t$, $\Sigma(J_1) \equiv t_1$ (mo

Assuming typical distributio
cost r, since $M \approx r$.

Generalized moduli

Choose M, t_1, r with $M \approx r$.
(Original moduli algorithm is the special case $r = 2^{n/4}$.)

Take set S_{11}, $\#S_{11} = r$, where $J_{11} \in S_{11} \Rightarrow J_{11} \subseteq \{1, \ldots, n/4\}$.
(Original algorithm: S_{11} is the set of *all* $J_{11} \subseteq \{1, \ldots, n/4\}$.)
Compute $\Sigma(J_{11}) \bmod M$ for each $J_{11} \in S_{11}$.

Similarly take a set S_{12} of r subsets of $\{n/4 + 1, \ldots, n/2\}$.
Compute $t_1 - \Sigma(J_{12}) \bmod M$ for each $J_{12} \in S_{12}$.

Find all collisions $\Sigma(J_{11}) \equiv t_1 - \Sigma(J_{12})$,
i.e., $\Sigma(J_1) \equiv t_1 \pmod{M}$
where $J_1 = J_{11} \cup J_{12}$.
Compute each $\Sigma(J_1)$.

Similarly S_{21}, $S_{22} \Rightarrow$
list of J_2 with $\Sigma(J_2) \equiv t - t_1$
$\Rightarrow$ each $t - \Sigma(J_2)$.

Find collisions $\Sigma(J_1) = t - \Sigma(J_2)$.

Success probability $r^4/2^n$ at finding any particular J with $\Sigma(J) = t$, $\Sigma(J_1) \equiv t_1 \pmod{M}$.

Assuming typical distribution: cost r, since $M \approx r$.

zed moduli

M, t_1, r with $M \approx r$.
l moduli algorithm
pecial case $r = 2^{n/4}$.)

: S_{11}, $\#S_{11} = r$, where
$_{11} \Rightarrow J_{11} \subseteq \{1, \ldots, n/4\}$.
l algorithm: S_{11} is the set
$_1 \subseteq \{1, \ldots, n/4\}$.)
e $\Sigma(J_{11})$ mod M
$J_{11} \in S_{11}$.

take a set S_{12} of r
of $\{n/4 + 1, \ldots, n/2\}$.
e $t_1 - \Sigma(J_{12})$ mod M
$J_{12} \in S_{12}$.

Find all collisions
$\Sigma(J_{11}) \equiv t_1 - \Sigma(J_{12})$,
i.e., $\Sigma(J_1) \equiv t_1 \pmod{M}$
where $J_1 = J_{11} \cup J_{12}$.
Compute each $\Sigma(J_1)$.

Similarly S_{21}, $S_{22} \Rightarrow$
list of J_2 with $\Sigma(J_2) \equiv t - t_1$
$\Rightarrow$ each $t - \Sigma(J_2)$.

Find collisions $\Sigma(J_1) = t - \Sigma(J_2)$.

Success probability $r^4/2^n$
at finding any particular J with
$\Sigma(J) = t$, $\Sigma(J_1) \equiv t_1 \pmod{M}$.

Assuming typical distribution:
cost r, since $M \approx r$.

Quantur

Capture
generaliz
as data s
$D(S_{11}, S$
Easy to
from S_{ij}

Convert
cost $r +$
$2^{0.2n}$ for

Use "am
to search
Total co

li

vith $M \approx r$.
lgorithm
$r = 2^{n/4}$.)

$_{11} = r$, where
$\{1, \ldots, n/4\}$.
n: S_{11} is the set
$.., n/4\}$.)
mod M
.

t S_{12} of r
$1, \ldots, n/2\}$.
J_{12}) mod M
^{2}.

Find all collisions
$\Sigma(J_{11}) \equiv t_1 - \Sigma(J_{12})$,
i.e., $\Sigma(J_1) \equiv t_1 \pmod{M}$
where $J_1 = J_{11} \cup J_{12}$.
Compute each $\Sigma(J_1)$.

Similarly S_{21}, S_{22} $\Rightarrow$
list of J_2 with $\Sigma(J_2) \equiv t - t_1$
$\Rightarrow$ each $t - \Sigma(J_2)$.

Find collisions $\Sigma(J_1) = t - \Sigma(J_2)$.

Success probability $r^4/2^n$
at finding any particular J with
$\Sigma(J) = t$, $\Sigma(J_1) \equiv t_1 \pmod{M}$.

Assuming typical distribution:
cost r, since $M \approx r$.

Quantum moduli (

Capture execution
generalized moduli
as data structure
$D(S_{11}, S_{12}, S_{21}, S$
Easy to move
from S_{ij} to adjace

Convert into quan
cost $r + \sqrt{r}2^{n/2}/$
$2^{0.2n}$ for $r \approx 2^{0.2n}$

Use "amplitude ar
to search for corre
Total cost $2^{0.3n}$.

r.

)

here

$n/4\}$.

the set

$2\}$.

M

Find all collisions
$\Sigma(J_{11}) \equiv t_1 - \Sigma(J_{12})$,
i.e., $\Sigma(J_1) \equiv t_1 \pmod{M}$
where $J_1 = J_{11} \cup J_{12}$.
Compute each $\Sigma(J_1)$.

Similarly S_{21}, $S_{22} \Rightarrow$
list of J_2 with $\Sigma(J_2) \equiv t - t_1$
$\Rightarrow$ each $t - \Sigma(J_2)$.

Find collisions $\Sigma(J_1) = t - \Sigma(J_2)$.

Success probability $r^4/2^n$
at finding any particular J with
$\Sigma(J) = t$, $\Sigma(J_1) \equiv t_1 \pmod{M}$.

Assuming typical distribution:
cost r, since $M \approx r$.

<u>Quantum moduli (0.3)</u>

Capture execution of
generalized moduli algorithm
as data structure
$D(S_{11}, S_{12}, S_{21}, S_{22})$.
Easy to move
from S_{ij} to adjacent T_{ij}.

Convert into quantum walk:
cost $r + \sqrt{r}2^{n/2}/r^2$.
$2^{0.2n}$ for $r \approx 2^{0.2n}$.

Use “amplitude amplificatio
to search for correct t_1.
Total cost $2^{0.3n}$.

Find all collisions
$\Sigma(J_{11}) \equiv t_1 - \Sigma(J_{12})$,
i.e., $\Sigma(J_1) \equiv t_1 \pmod{M}$
where $J_1 = J_{11} \cup J_{12}$.
Compute each $\Sigma(J_1)$.

Similarly S_{21}, $S_{22} \Rightarrow$
list of J_2 with $\Sigma(J_2) \equiv t - t_1$
$\Rightarrow$ each $t - \Sigma(J_2)$.

Find collisions $\Sigma(J_1) = t - \Sigma(J_2)$.

Success probability $r^4/2^n$
at finding any particular J with
$\Sigma(J) = t$, $\Sigma(J_1) \equiv t_1 \pmod{M}$.

Assuming typical distribution:
cost r, since $M \approx r$.

Quantum moduli (0.3)

Capture execution of
generalized moduli algorithm
as data structure
$D(S_{11}, S_{12}, S_{21}, S_{22})$.
Easy to move
from S_{ij} to adjacent T_{ij}.

Convert into quantum walk:
cost $r + \sqrt{r}2^{n/2}/r^2$.
$2^{0.2n}$ for $r \approx 2^{0.2n}$.

Use "amplitude amplification"
to search for correct t_1.
Total cost $2^{0.3n}$.

collisions

$\equiv t_1 - \Sigma(J_{12})$,

$_1) \equiv t_1 \pmod{M}$

$_1 = J_{11} \cup J_{12}$.

e each $\Sigma(J_1)$.

S_{21}, $S_{22} \Rightarrow$

$_2$ with $\Sigma(J_2) \equiv t - t_1$

$t - \Sigma(J_2)$.

lisions $\Sigma(J_1) = t - \Sigma(J_2)$.

probability $r^4/2^n$

g any particular J with

t, $\Sigma(J_1) \equiv t_1 \pmod{M}$.

g typical distribution:

since $M \approx r$.

Quantum moduli (0.3)

Capture execution of generalized moduli algorithm as data structure $D(S_{11}, S_{12}, S_{21}, S_{22})$.
Easy to move from S_{ij} to adjacent T_{ij}.

Convert into quantum walk: cost $r + \sqrt{r}2^{n/2}/r^2$.
$2^{0.2n}$ for $r \approx 2^{0.2n}$.

Use "amplitude amplification" to search for correct t_1.
Total cost $2^{0.3n}$.

Quantur

Central
Combine
with "re
2010 Ho
Subset-s
new reco

Lower-le
Ambaini
"combin
and a sk
history-i
We use
Much ea

$J_{12})$,
(mod M)
J_{12}.
$J_1)$.

$\Rightarrow$
$J_2) \equiv t - t_1$

.

$J_1) = t - \Sigma(J_2)$.

$y\ r^4/2^n$
ticular J with
$\equiv t_1 \pmod{M}$.

distribution:
r.

Quantum moduli (0.3)

Capture execution of
generalized moduli algorithm
as data structure
$D(S_{11}, S_{12}, S_{21}, S_{22})$.
Easy to move
from S_{ij} to adjacent T_{ij}.

Convert into quantum walk:
cost $r + \sqrt{r}2^{n/2}/r^2$.
$2^{0.2n}$ for $r \approx 2^{0.2n}$.

Use "amplitude amplification"
to search for correct t_1.
Total cost $2^{0.3n}$.

Quantum reps (0.2

Central result of t
Combine quantum
with "representati
2010 Howgrave-G
Subset-sum expon
new record.

Lower-level impro
Ambainis uses ad-
"combination of a
and a skip list" to
history-independe
We use radix tree
Much easier, pres

$_1$

$\Sigma(J_2)$.

vith

d M).

n:

Quantum moduli (0.3)

Capture execution of generalized moduli algorithm as data structure $D(S_{11}, S_{12}, S_{21}, S_{22})$.
Easy to move from S_{ij} to adjacent T_{ij}.

Convert into quantum walk:
cost $r + \sqrt{r}2^{n/2}/r^2$.
$2^{0.2n}$ for $r \approx 2^{0.2n}$.

Use "amplitude amplification" to search for correct t_1.
Total cost $2^{0.3n}$.

Quantum reps (0.241 . . .)

Central result of the paper:
Combine quantum walk with "representations" idea
2010 Howgrave-Graham–Jou
Subset-sum exponent 0.241
new record.

Lower-level improvement:
Ambainis uses ad-hoc "combination of a hash tabl and a skip list" to ensure history-independence.
We use radix trees.
Much easier, presumably fas

Quantum moduli (0.3)

Capture execution of generalized moduli algorithm as data structure $D(S_{11}, S_{12}, S_{21}, S_{22})$. Easy to move from S_{ij} to adjacent T_{ij}.

Convert into quantum walk: cost $r + \sqrt{r}2^{n/2}/r^2$. $2^{0.2n}$ for $r \approx 2^{0.2n}$.

Use "amplitude amplification" to search for correct t_1. Total cost $2^{0.3n}$.

Quantum reps (0.241...)

Central result of the paper: Combine quantum walk with "representations" idea of 2010 Howgrave-Graham–Joux. Subset-sum exponent 0.241...; new record.

Lower-level improvement: Ambainis uses ad-hoc "combination of a hash table and a skip list" to ensure history-independence. We use radix trees. Much easier, presumably faster.