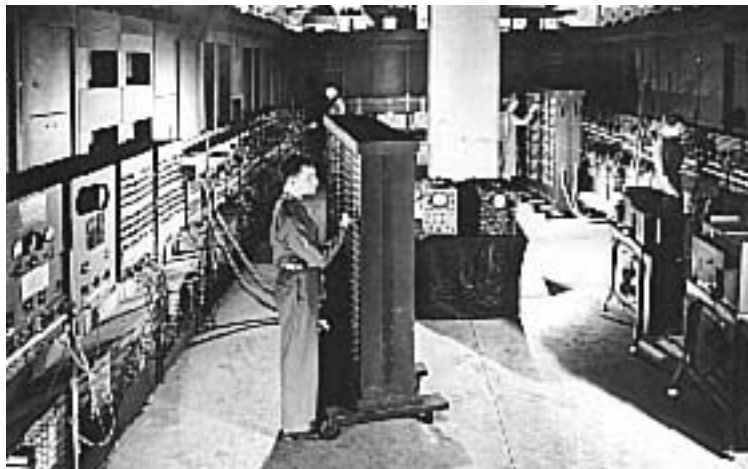


How the ENIAC took a Square Root

Brian J. Shelburne
Dept of Math & Comp Sci
Wittenberg University
April 6, 2002

bshelburne@wittenberg.edu

<http://www4.wittenberg.edu/academics/mathcomp/shelburn.shtml>



Why is this interesting?

Taking a square root is hard to do.

ENIAC was a “primitive” device.

Only two early computers (Z3 and ENIAC)
implemented a square root operation

Overview of ENIAC

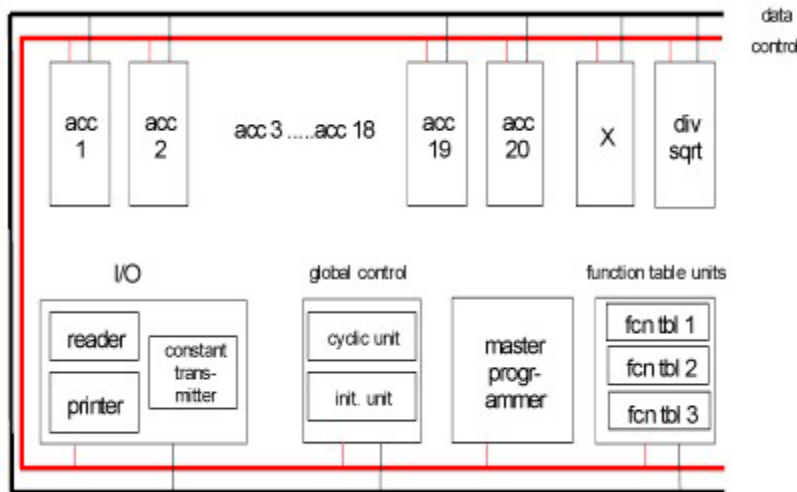
Not a “stored program” computer; memory limited to data

Twenty 10-decimal digit (plus sign) accumulators for addition and subtraction (storage & processing not separate)

High-speed multiplier computed partial products. Accumulators used for their summation

Divider/Square Rooter: orchestrated a series of operations using 3 to 6 accumulators to divide or take square root.

Control distributed – ENIAC programmed by “wiring” together units via data and control buses to perform calculations



Functional Diagram of ENIAC after Fig 4. from “The ENIAC: History, Operation and Reconstruction in VLSI” – see references

40 panels each 2 feet wide by 8 feet high arranged in a U-shape in a 30 by 60 foot room. 17,000+ vacuum tubes!

A method for approximating the square root of an integer m

Observe: The sum of the first n odd integers is n^2 .

$$1 + 3 + \dots + (2n - 1) = \sum_{i=1}^n (2i - 1) = n^2$$

Therefore if n is the *smallest integer* such that $m - \sum_{i=1}^n (2i - 1) < 0$

then $(n-1)^2 \leq m < n^2$ or $n-1 \leq \sqrt{m} < n$ or if $a = 2n-1$ is the n^{th} odd integer then

$$\frac{a-1}{2} \leq \sqrt{m} < \frac{a+1}{2}$$

For additional precision multiply m by the k^{th} power of 100, find the square root of $m \times 100^k$ then divide the answer by k^{th} power of 10 since $\sqrt{m \times 100^k} = \sqrt{m} \times 10^k$.

Example: Find $\sqrt{7251}$ to 2 digits below the decimal point

$$7251 \times 100^2 = 72,510,000$$

$$72,510,000 - (1 + 3 + 5 + \dots + 17031) = -12256.$$

Therefore $a = 17031$ and

$$\frac{17031-1}{2} \leq \sqrt{72,510,000} < \frac{17031+1}{2} \text{ or } 8515 \leq \sqrt{72,510,000} < 8516$$

After dividing by 10^2 , $85.15 \leq \sqrt{7251} < 85.16$.

(real answer $\sqrt{7251} = 85.15280383$)

Reality Check: The ENIAC could do 5000 addition/subtractions per second! Therefore to perform 2×8516 additions and subtractions would take approximately 3.4 seconds!

A More Efficient Way – But at a Price!

The sum of the first n odd multiples of 100 is a square.

$$100 + 300 + \dots + (2n - 1) \times 100 = \sum_{i=1}^n (2i - 1) \times 100 = n^2 \times 100.$$

Therefore if $(2n-1) \times 100$ is the *smallest* odd multiple of 100 such

that $m - \sum_{i=1}^n (2i - 1) \times 100 < 0$, then $(n-1)^2 \times 100 \leq m < n^2 \times 100$ or

$$(n-1) \times 10 \leq \sqrt{m} < n \times 10!$$

Thus with *one tenth* of the work we can find \sqrt{m} to the nearest *tens*.

However every odd multiple of 100 is the sum of 10 consecutive odd integers:

$$\begin{aligned} (2n-1) \times 100 &= 200(n-1) + 100 = \\ &10 \times 20(n-1) + (1 + 3 + 5 + \dots + 19) = \\ &[20(n-1) + 1] + [20(n-1) + 3] + \dots + [20(n-1) + 19] \end{aligned}$$

Or letting $N = (2n-1) \times 100$, can rewrite its sum as follows

$$N = [N/10 - 9] + [N/10 - 7] + \dots + [N/10 + 9]$$

Thus we can add back the odd integers starting with $N/10 + 9$ until the sign is positive. If $N/10 \pm j$ (j an odd integer between 1 and 9) was the *last* (largest) odd integer added back, then $N/10 \pm j$ is also the *smallest* odd integer such that $m - (1 + 3 + \dots + (N/10 \pm j)) < 0$.

Example: Find $\sqrt{72,150,000}$

1. Subtract the odd multiples of 100

$$72,150,000 - (100 + 300 + \dots + 170100) = 89900 \text{ and}$$
$$72,150,000 - (100 + 300 + \dots + 170300) = -80400 \text{ so } N = 170300.$$

The largest odd integer subtracted was $170300/10 + 9 = 17039$.

2. Add back the decreasing sequence of odd integers starting with 17039

$$-80400 + (17039 + 17037 + \dots + 17033) = -12256 \text{ and}$$
$$-80400 + (17039 + 17037 + \dots + 17033 + 17031) = 4775.$$

Thus $N = 17031$ and which agrees with the previous result.

Reality Check: With a reduction in the number of additions and subtractions by a factor of 10, ENIAC could calculate the square root in approximately 0.34 seconds! Note that division by 10 on the ENIAC can be accomplished by a simple right shift!

General Results

Result #1: For $k \geq 0$ the sum of the first n odd multiples of 100^k is a square. That is

$$\sum_{i=1}^n (2i-1) \times 100^k = n^2 \times 100^k$$

Therefore if n is the *smallest* odd multiple of 100^k such that

$$m - \sum_{i=1}^n (2n-1) \times 100^k < 0 \text{ then } (n-1) \times 10^k \leq \sqrt{m} < n \times 10^k$$

Reality Check! For $k > 0$ this is a terrible approximation for \sqrt{m} !

Result #2: For $k > 0$, if $N = (2n-1) \times 100^k$ is any odd multiple of 100^k then N is the sum of ten consecutive odd multiples of 100^{k-1} . Specifically

$$N = \left(\frac{N}{10} - 9 \times 100^{k-1}\right) + \left(\frac{N}{10} - 7 \times 100^{k-1}\right) + \dots + \left(\frac{N}{10} + 9 \times 100^{k-1}\right)$$

Proof: Do the algebra! The $k = 1$ case says that any odd multiple of 100 is the sum of ten consecutive odd integers.

The Algorithm

1. Starting with 100^k , **subtract** the increasing sequence of odd multiples of 100^k from m until a negative result is obtained.

If $N = (2n-1) \times 100^k$ was the last multiple subtracted, then N is the sum of ten consecutive odd multiples of 100^{k-1} .

2. $N/10 + 9 \times 100^{k-1}$ was the last (largest) power of 100^{k-1} subtracted. Starting with this value, **add back** the decreasing sequence of odd multiples of 100^{k-1} until a positive result is obtained.

Replace $k-1$ by k and let N be the last multiple of 100^k (remember $k-1$ is now k) added back. If $k = 0$ then you're done so go to step 4; else N is the sum of ten consecutive odd multiples of 100^{k-1} so continue on to step 3

3. $N/10 - 9 \times 100^{k-1}$ was the last (smallest) power of 100^{k-1} added back. Starting with this value, **subtract** the increasing sequence of odd multiples of 100^{k-1} until a negative result is obtained.

Replace $k-1$ by k and let N be the last multiple of 100^k (remember $k-1$ is now k) subtracted. If $k = 0$ then you're done so go to step 4; else N is the sum of ten consecutive odd multiples of 100^{k-1} so go back to step 2.

4. N is the largest odd integer such that

$$m - (1 + 3 + \dots + N) < 0.$$

Thus $N-1 \leq 2\sqrt{m} < N+1$ or $(N-1)/2 \leq \sqrt{m} < (N+1)/2$.

The ENIAC used N for twice the square root

Example: Find $\sqrt{72,510,000}$ (again!)

Step #1 - Subtract

$$72,510,000 - (1 \times 100^3 + 3 \times 100^3 + \dots + 17 \times 100^3) = -8,490,000$$

$$17 \times 100^3 / 10 + 9 \times 100^2 = 179 \times 100^2$$

Step #2 – Add Back

$$-8,490,000 + (179 \times 100^2 + 177 \times 100^2 + \dots + 171 \times 100^2) = 260,000$$

$$171 \times 100^2 - 9 \times 100 = 1701 \times 100$$

Step #3 - Subtract

$$260,000 - (1701 \times 100 + 1703 \times 100) = -80400$$

$$1703 \times 100 / 10 + 9 = 17039$$

Step #4 – Add Back

$$-80400 + (17039 + 17037 + \dots + 17031) = 4775$$

Thus $N = 17031$ which agrees with previous results

Reality Check: This required approximately 42 addition/subtractions meaning the ENIAC could have done this in approximately 0.0084 sec.

How the ENIAC took the Square Root of $m = 72,510,000$

The ENIAC divider/square rooter actually computed *twice* the square root of m . It also *scaled* its calculations to achieve an additional 4 digits of precision! The ENIAC loaded m into one accumulator called the **numerator** and set a second accumulator called the **denominator** to 100^4 by putting a 1 in the 9th position.

Step 1: Subtract

Numerator: 0,072,510,000	Denominator: 0,100,000,000
-0,027,490,000	0,300,000,000

Right shift denominator and add 9 to 7th digit ?

Numerator: -0,027,490,000	Denominator: 0,019,000,000
---------------------------	----------------------------

No! *Scale* by left shifting the numerator and by not shifting the denominator

Numerator: -0,274,900,000	Denominator: 0,300,000,000
Subtract 11 from 8 th and 9 th digits ->	0,190,000,000

Step 2: Add back

Numerator: -0,274,900,000	Denominator: 0,190,000,000
.
0,085,100,000	0,150,000,000

Step 3: Subtract

Numerator: 0,851,000,000	Denominator: 0,121,000,000
.
-0,145,000,000	0,173,000,000

Step 4: Add back

Numerator: -1,450,000,000	Denominator: 0,171,900,000
.
0,089,900,000	0,170,100,000

jump ahead to Step 9: Subtract

Numerator: 0,651,600,000	Denominator: 0,170,305,601
0,481,854,399	0,170,305,603
0,311,548,796	0,170,305,605
0,141,243,191	0,170,305,607
-0,029,062,416	0,170,305,609

Thus 17030.5607 after scaling is the best approximation to twice $\sqrt{72,510,000}$ which is 8515.28035 (actual value is 8515.280383)

Summary

That wasn't so bad!

Works for small numbers like $\sqrt{2}$

Step 1: Subtract

Numerator:	0,000,000,002	Denominator:	0,100,000,000
	-0,099,999,998		0,300,000,000

Step 2: Add back

Numerator:	-0,999,999,980	Denominator:	0,190,000,000

	0,000,000,020		-0,010,000,000

Step 9: Subtract

Numerator:	0,000,060,400	Denominator:	0,000,028,281
	0,000,032,119		0,000,028,283
	0,000,003,836		0,000,028,285
	-0,000,024,449		0,000,028,287

2.8285 is a good approximation for twice $\sqrt{2}$

Reality Check: According to published specs the ENIAC could take the square root of a 9 digit number in about 26 msec on average (0.026 sec)

References

<http://www4.wittenberg.edu/academics/mathcomp/bjsdir/ENIACSquareRoot.htm>

ENIAC: The Triumph and Tragedies of the World's First Computer; Scott McCartney

“The ENIAC: History, Operation and Reconstruction in VLSI” by Jan Van der Spiegel et al. *from The First Computers: History and Architectures* edited by R. Rojas and U. Hashagen