Programming the Old School Algorithm for Calculating Square Root
by Gerald R. Rising
SUNY Distinguished Teaching Professor Emeritus
University at Buffalo

Since the old school square root procedure introduced in Chapter 4 of *Inside Your Calculator* is an algorithm, we can develop a program to carry it out. It turns out that the program is more complicated than the one for Newton’s Method, which appears in that chapter, but developing that program should give further insights into the translation of a formal algorithm into a program.

Here is the algorithm we wish to program, slightly modified from that chapter where the mathematics of the process is explained and justified.

1. Separate the radicand into pairs of digits working left and right from the decimal point. Place a decimal point where your answer will appear above the original decimal point.
2. Find the nearest square less than or equal to the leftmost pair.
3. Write this square beneath the leftmost pair and its square root above it as in the long division algorithm.
4. Subtract your square from the radicand pair as in long division.
5. Bring down the next pair of digits.
6. Multiply your partial answer by 20 and write the product to the left of the result you obtained from steps 4 and 5 (or later from step 9).
7. Divide your result in step 5 (or 9) by your answer in step 6, writing the quotient as the next digit in your answer.
8. Add this digit to your answer in step 6 and then multiply your result by the same digit, placing the product under your answer from step 5.
9. Subtract and continue from step 5 until the number of digits required is reached.

Notice that in this algorithm the first four steps are different from the remaining steps. Steps 1-4 are only processed once; steps 5-9 are repeated until the task is completed to the desired accuracy. It seems natural then to address these two groups separately.

Algorithm Steps 1-4

First we will need to get our problem defined. We will seek \( \sqrt{N} \) to D digit accuracy, so we enter this information in our program with the command: *Prompt N,D*.

The second step in the algorithm asks us to find the nearest square less than that first pair of digits on the left. To facilitate doing this it will be convenient if we shift the digits of N (temporarily) to the right or left two decimal places at a time to give us a new value, M, between 1 and 100, that is with \( 1 \leq M < 100 \).\(^1\) For example, if our value of N is 1927 we will shift the decimal to the left two places to produce \( M = 19.27 \). Similarly, if our original number had been .00001927, we would have shifted the decimal to the right six places to produce \( M = 19.27 \). Clearly, this process will involve multiplying or dividing by 100.

It is important for you to see that the digits in numbers like \( \sqrt{1927} \), \( \sqrt{192700} \), \( \sqrt{1927} \) and \( \sqrt{0.000001927} \) are the same so long as they are multiples by 100 of each other. You can assure yourself of this fact with your calculator.\(^2\) To continue with a single example, we will follow the calculation of \( \sqrt{1927} \), that is, \( N = 1927 \), through the remainder of this essay. We will seek to calculate this square root to 3 digit accuracy, thus we have D = 3.

\(^1\)We will retain the value of N to report at the end of the program, modifying M as the program progresses.

\(^2\)When you do this, it would be well to determine \( \sqrt{1927} \) also, to show that the digits in this number differ from those others. In this case the radicand differs by a factor of 10, not 100, as required. The general rule is: The digits of \( \sqrt{M} \) are the same as the digits of \( \sqrt{M \times 100^n} \) for any integer n. The reason for this: \( \sqrt{M \times 100^n} = 10^n \sqrt{M} \).
Here are program steps that will adjust M so that $1 \leq M < 100$:

\[
\begin{align*}
N &\rightarrow M \\
0 &\rightarrow B \\
0 &\rightarrow S \\
\text{While } M < 1 & \\
M \times 100 &\rightarrow M \\
S + 1 &\rightarrow S \\
\text{End} \\
\text{While } M > 100 & \\
M / 100 &\rightarrow M \\
B + 1 &\rightarrow B \\
\text{End}
\end{align*}
\]

We now have S representing the number of pairs of digits shifted to make a small number larger and B, the number of pairs of digits shifted to make a big number smaller. We will use these values at the end of the program to readjust the decimal point location in the final square root. In our example, the second While loop would change $M = 1927$ to $M = 19.27$, in the process making $B = 1$.

Now we are prepared to carry out algorithm steps 2-4. In what follows we will let $X$ represent the answer we will be building. Here are a series of program commands that will carry out steps 2-3 of the algorithm:

\[
\begin{align*}
2 &\rightarrow X \\
\text{While } X \times X < M & \\
X + 1 &\rightarrow X \\
\text{End} \\
X - 1 &\rightarrow X
\end{align*}
\]

Let’s see what is going on in those program lines. At this point in our example $M = 19.27$ and we know that, since $4^2 = 16$ and $5^2 = 25$, we have $4 < \sqrt{19.27} < 5$ so our answer will be 4 point something. Thus the integer value we seek is 4. To arrive at the number we begin with $X = 2$ in the first of these lines. Clearly $2^2 = 4 < 19.27$ so we enter the While loop of the second line and $X$ is increased to 3 in the third line. In the same way $3^2 = 9 < 19.27$ and the loop is repeated. Now $X = 4$ and $4^2 = 16 < 19.27$ and we increase $X$ to 5. Since $5^2 = 25 > 19.27$, we exit the While with $X = 5$ and skip to the 5th of these lines. This value of $X$ is, of course, one too large. Since it will always be one too large after this series of program lines (including if we had started with $M = 1$), the value is corrected in the final line and we are left with the desired value, $X = 4$.

We still have that fourth step to take care of. We must subtract the square of $X$ from the value of $M$. This is easily accomplished in the next program line: $M - X \times X \rightarrow M$.

Notice one thing about this value of $M$, however. At this point in the algorithm $M$ is only a one or two digit number; however, our calculated $M$ includes all of the remaining digits. For our example, $M$ is now $19.27 - 16 = 3.27$. (Other examples might have still more digits.) $M$ now represents all of the value left to be processed, whereas the algorithm value would be just 3, before the next digits are “brought down.” What corresponds to the algorithm value is the integer value of $M$. Each time we multiply the calculated $M$ by 100 the integer result will correspond to the new value in the algorithm with the next “pair of digits brought down.”

---

3In the following programs some lines are indented to show structure. Do not use this spacing when entering these programs in a calculator. Also recall that the $\rightarrow$ is the “store” or $\text{STO}>$ instruction.
We are now ready to proceed to the algorithm steps that follow.

**Algorithm Steps 5-9**

As I pointed out earlier, algorithm steps 5-9 are to be repeated until we reach the desired number of digits of accuracy in the square root, X. We will keep track of the number of those digits already calculated with the variable, I. At this point in the program I = 1, because we calculated the first digit of X in steps 1-4. We will want to find an extra digit for X in order to round our answer at the end of the program. Thus we will carry out algorithm steps 5-9 so long as I < D+1. (When I = D+1 we will have processed D+1 digits.) A *While* loop will accomplish this.

Here are the program lines for this *While* loop:

```
1→I
While I<D+1
    M*100→M
    X*10→X
    X*2→Y
    int(M/Y)→T
    While (Y+T)*T>M
        T-1→T
    End
    M-(Y+T)*T→M
    X+T→X
    I+1→I
End
```

Clearly a great deal is going on in this loop and, although it accomplishes steps 5-9, it is not always easy to recognize how it is doing that. For that reason we’ll take those lines in small bites.

First we have that algorithm step 5: “Bring down the next pair of digits.” This is where we multiply our value of M by 100 in the program line `M*100→M`. M was 3.27; after this line M = 327.

Now we have that strange algorithm step 6: “Multiply your partial answer by 20 and write the product to the left of the result you obtained from steps 4 and 5 (or later from step 9).” Multiplying by 20 is, of course, the same as multiplying by 10 and then by 2. By breaking this instruction down into those two parts, we can first modify X to make room for the next digit in the answer with the line `X*10→X`. In our example, this prepares our partial answer, X = 4, to append another digit by multiplying it by 10 to make X = 40. The next calculated digit, which will turn out to be 3, when added to this, will give X = 43. The decimal will be placed in this number later.

We still have to multiply the current value of X by two. We do so in the next line, `X*2→Y`. The resulting Y is that number “written to the left of” M. In our example Y = 2 * 40 = 80.

We now divide M by this Y, taking the integer value of the result: `int(M/Y)→T`. In our example the program would calculate T = int(327/80) = 4.

This is designated T, because it is a trial result. Remember that the algorithm calls for us to add T to M before multiplying by it. That will often make the product larger than M. In fact, in our example, when we form the product (Y + T)T, we have (80 + 4)4 = 336, which is greater than M = 327. Since we must deal only with positive numbers in this process, we must reduce T. That is the reason for the *While* loop that follows the initial calculation of T:
While \((Y+T)\times T > M\)
\[ T-1 \rightarrow T \]
End

In our example, we now have \(T = 4 - 1 = 3\), and the product \((Y + T)T\) is \((80 + 3)3 = 249\), which is clearly less than \(M = 327\). With the adjusted \(T\) value we can proceed with the subtraction that follows in algorithm step 9. We do this with the following instruction: \(M - (Y+T) \times T \rightarrow M\), with \(T\) also added to our already prepared value of \(X\) in the line: \(X+T \rightarrow X\). At this point we have processed one more digit so we increment \(I\) in the line: \(I+1 \rightarrow I\). After this first pass through that large \texttt{While} loop, the values in our example would be \(X = 43\), \(M = 327 - 249 = 78\) and \(I = 2\).

When, after additional passes, we finally emerge from this loop,\(^4\) we will have completed the algorithm steps; however, we will usually not have the decimal point correctly placed and we will also have one more digit than we desire in our final answer. For our calculation of \(\sqrt{1927}\) to three digit accuracy (that is, \(D = 3\)), the value \(X = 4389\) must be adjusted to give the final answer, 43.9.

**Rounding and Placing the Decimal Point**

We address the problem of rounding first. At this point we have an integer that we wish to round to the nearest ten. Thus, for the desired three digit accuracy of our example, we would round that 4389 to 4390, since in 4389 the units digit, \(9 \geq 5\). To do this, we must pick off that units digit, the 9 of our example, and compare it with 5, deciding by that comparison how to modify \(X\). Here are the program lines that accomplish this:

\[
\begin{align*}
X-10 \times \text{int}(X/10) & \rightarrow R \\
\text{If } R \geq 5 \\
X+10 & \rightarrow X
\end{align*}
\]

In the first of those lines, the remainder of the integer division of \(X\) by 10 is calculated. For our continuing example, the remainder, \(R\), when 4389 is divided by 10 is exactly that 9 we are looking for. In the second line, we compare that \(R\) with 5. In our example \(9 \geq 5\), so \(X\) is increased by 10 to produce the desired \(X = 4390\).

That next program line is \(\text{int}(X/10) \rightarrow X\). Notice how that line simply “throws away” that last digit, producing in our example, \(X = 439\), the three correct digits that we sought when we set \(D = 3\).

Now we must place the decimal point based on the changes we made back at the beginning of the program. We do this in two steps. First we move the decimal point back to the form we used after we first adjusted \(N\) to become \(M\). Recall that in that form, we had \(1 \leq M < 100\). Since we are calculating \(X = \sqrt{M}\), we will want \(1 \leq X < 10\). This is done in the single program line, \(X/10 \land (D-1) \rightarrow X\). With \(D = 3\) in our example, this calculates 439/10², which gives us 4.39.

Next we must move that decimal point the correct number of places. Recall that, early in the program, we multiplied by 100 \(S\) times or divided by 100 \(B\) times. Our answer is adjusted by dividing by 10 \(S\) times and multiplying by 10 \(B\) times. That is done in the program line \(X \times 10 \land (B-S) \rightarrow X\). In our example, we changed \(N = 1927\) to \(M = 19.27\) by dividing by 100, thus \(B = 1\) while \(S\) remains 0. This program line then calculates \(4.39 \times 10^{1-0}\) to produce \(X = 43.9\), which is the square root we desire to three digit accuracy.

\(^4\)In our example, we would pass through this loop three times giving the successive results 43, 438 and finally 4389. After those passes both \(I\) and \(D+1\) would be 4 so the calculation would exit the \texttt{While} loop. Tracing the program to see this processing is a useful exercise.
Finally, the results are produced with the program line, Disp N,X, which displays the original N and the value of its square root to D digit accuracy.
Here then is the final program:

```
PROGRAM:SCHSQRT
Prompt N,D
N→M
0→B
0→S
While M<1
   M*100→M
   S+1→S
End
While M>100
   M/100→M
   B+1→B
End
2→X
While X*X<M
   X+1→X
End
X-1→X
M-X*X→M
1→I
While I<D+1
   M*100→M
   X*10→X
   X*2→Y
   int(M/Y)→T
   While (Y+T)*T>M
      T-1→T
   End
   M-(Y+T)*T→M
   X+T→X
   I+1→I
End
X-10*int(X/10→R
If R≥5
   X+10→X
   int(X/10)→X
   X/10^(D-1)→X
   X*10^(B-S)→X
Disp N,X
```

The program may be downloaded from the website www.buffalo.edu/~insrisg/InsideYourCalculator.htm.