Reciproot Algorithm—Correctly Rounded? *

Ren-Cang Li
Department of Mathematics
University of California at Berkeley
Berkeley, California 94720

September 6, 1992

Computer Science Division Technical Report UCB//CSD-94-850, University of California, Berkeley, CA 94720, December, 1994.

Abstract

This note attempts to give a detailed error analysis of *Reciproot Algorithm* proposed by Kahan and Ng in 1986. It is showed that the algorithm yields correctly rounded square root under all rounding modes.

1 Initial Approximation

Let x_0 and x_1 be the leading and the trailing 32-bit words of a floating point number x (in IEEE double format) respectively

*This material is based in part upon work supported by Argonne National Laboratory under grant No. 20552402 and the University of Tennessee through the Advanced Research Projects Agency under contract No. DAAL03-91-C-0047, by the National Science Foundation under grant No. ASC-9005933, and by the National Science Infrastructure grants No. CDA-8722788 and CDA-9401156.

	1	11	52
x:	s	e	f

	1	11	20
x_0:	s	e	f_1

	32
x_1:	f_2

By performing shifts and subtracts on x_0 and y_0 (both regarded as integers), we obtain a 7-bit approximation of $1/\sqrt{x}$ as follows.

$$\begin{aligned} k &:= \texttt{0x5fe80000} - (x_0 \gg 1); \\ y_0 &:= k - T2[63\&(k \gg 14)]. \end{aligned}$$

Here k is a 32-bit integer and $T2[\cdot]$ is an integer array containing correction terms. Now magically the floating value of y (y's leading 32-bit word is y_0, the value of its trailing word y_1 is set to zero) approximates $1/\sqrt{x}$ to more than 7-bit.

Value of $T2[64]$ are

This table needs to be checked!

```
0x1500, 0x2ef8, 0x4d67, 0x6b02, 0x87be, 0xa395, 0xbe7a, 0xd866,
0xf14a, 0x1091b,0x11fcd,0x13552,0x14999,0x15c98,0x16e34,0x17e5f,
0x18d03,0x19a01,0x1a545,0x1ae8a,0x1b5c4,0x1bb01,0x1bfde,0x1c28d,
0x1c2de,0x1c0db,0x1ba73,0x1b11c,0x1a4b5,0x1953d,0x18266,0x16be0,
0x1683e,0x179d8,0x18a4d,0x19992,0x1a789,0x1b445,0x1bf61,0x1c989,
0x1d16d,0x1d77b,0x1dddf,0x1e2ad,0x1e5bf,0x1e6e8,0x1e654,0x1e3cd,
0x1df2a,0x1d635,0x1cb16,0x1be2c,0x1ae4e,0x19bde,0x1868e,0x16e2e,
0x1527f,0x1334a,0x11051,0xe951, 0xbe01, 0x8e0d, 0x5924, 0x1edd.
```

Now we prove our claimed accuracy of y as an initial approximation to $1/\sqrt{x}$. As a matter of fact, x can be written as $x = (-1)^s 2^e \times 1.f = 2^e \times 1.f$. Tedious analysis shows that

$$k \sim 2^{-\frac{e-1}{2}-1}\left(1 + 1 - \frac{1.f_1'}{2}\right), \quad \text{if} \quad e \quad \text{is} \quad \text{odd}, \tag{1}$$

$$k \sim 2^{-\frac{e}{2}-1}\left(1 + \frac{3 - 1.f_1'}{2}\right), \quad \text{if} \quad e \quad \text{is} \quad \text{even}, \tag{2}$$

where f_1' is f_1 except its last bit is set to zero.

Case (i): e is odd and bits of f_1' are not all zero.

Denote $\alpha = 1.f_1'$. Since $1 \le \alpha \le 2 - 2^{-19}$, we have $2^{-20} \le 1 - \frac{\alpha}{2} \le 2^{-1}$. Assume for the moment that $\alpha > 1$, then $1 - \frac{\alpha}{2} < 2^{-1}$. Write

$$1 - \frac{\alpha}{2} = 0.0a_2a_3a_4a_5a_6 \cdots a_{21}.$$

$63\&(k \gg 14) = 0a_2a_3a_4a_5a_6$ which gives us valuable information to construct correction terms. Let

$$t = 0.0a_2a_3a_4a_5a_6, \quad \epsilon = 0.000001_2 = 2^{-6}.$$

Now we know that $t \le 1 - \frac{\alpha}{2} < t + \epsilon$, which produces

$$2(1-t) - 2\epsilon < \alpha \le 2(1-t).$$

(if $t = 0$, we have $2(1-\epsilon) < \alpha < 2$.)

We are required to choose a number d associated with those information so that

$$1 + 1 - \frac{\alpha}{2} - d \quad \text{approximates} \quad \frac{2}{\sqrt{2 \times 1.f}} \quad \text{accurately} \quad \text{enough.}$$

As this approximation could not be correct to more than 20-bits, we can restate it as

$$1 + 1 - \frac{\alpha}{2} - d \quad \text{approximates} \quad \frac{2}{\sqrt{2\alpha}} \quad \text{accurately} \quad \text{enough.}$$

Elementary arguments show that the best d is

$$\begin{aligned} d &= \frac{1 + t + \epsilon - \frac{2}{\sqrt{2(2(1-t-\epsilon))}} + 1 + t - \frac{2}{\sqrt{2(2(1-t))}}}{2} \\ &= \frac{1 + t + \epsilon - \frac{1}{\sqrt{1-t-\epsilon}} + 1 + t - \frac{1}{\sqrt{1-t}}}{2} \\ &= 1 + t + \frac{\epsilon}{2} - \frac{1}{2}\left(\frac{1}{\sqrt{1-t}} + \frac{1}{\sqrt{1-t-\epsilon}}\right), \end{aligned}$$

except possibly for one case when the interval $[2(1-t-\epsilon), 2(1-t)]$ contains $\sqrt[3]{2}$, for which $t = 0.010111_2 = 0.359375$. Individual check shows that this gives no trouble. To find the largest error in using $1+1-\frac{\alpha}{2}-d$ to approximate $\frac{2}{\sqrt{2\alpha}}$, it suffices for us to look at

$$\begin{aligned}
&1+t-d-\frac{2}{\sqrt{2(2(1-t))}} \\
&\quad= -\frac{\epsilon}{2}+\frac{1}{2}\left(\frac{1}{\sqrt{1-t-\epsilon}}-\frac{1}{\sqrt{1-t}}\right) \\
&\quad= -\frac{\epsilon}{2}\left(1-\frac{1}{(1-t-\epsilon)\sqrt{1-t}+(1-t)\sqrt{1-t-\epsilon}}\right),
\end{aligned}$$

whose absolute value for all possible $0 \le t \le 2^{-1}$ is less than or equal to $\frac{\epsilon}{2} \times 0.4941 < \frac{\epsilon}{4} = 2^{-8}$.

Case (ii): e is even and bits of f_1' are not all zero.

Denote $\alpha = 1.f_1'$. Since $1 \le \alpha \le 2 - 2^{-19}$, we have $2^{-1} + 2^{-20} \le \frac{3-\alpha}{2} \le 1$. Assume for the moment that $\alpha > 1$, then $\frac{3-\alpha}{2} < 1$. Write

$$\frac{3-\alpha}{2} = 0.1a_2a_3a_4a_5a_6\cdots a_{21}.$$

$63\&(k \gg 14) = 1a_2a_3a_4a_5a_6$ which gives us valuable information to construct correction terms. Let

$$t = 0.1a_2a_3a_4a_5a_6, \quad \epsilon = 0.000001_2 = 2^{-6}.$$

Now we know that $t \le \frac{3-\alpha}{2} < t+\epsilon$, which produces

$$3-2(t+\epsilon) < \alpha \le 3-2t.$$

We are required to choose a number d associated with those information so that

$$1+\frac{3-\alpha}{2}-d \quad \text{approximates} \quad \frac{2}{\sqrt{1.f}} \quad \text{accurately} \quad \text{enough.}$$

As this approximation could not be correct to more than 20-bits, we can restate it as

$$1+\frac{3-\alpha}{2}-d \quad \text{approximates} \quad \frac{2}{\sqrt{\alpha}} \quad \text{accurately} \quad \text{enough.}$$

Elementary arguments show that the best d is

$$\begin{aligned} d &= \frac{1+t+\epsilon-\frac{2}{\sqrt{3-2(t+\epsilon)}}+1+t-\frac{2}{\sqrt{3-2t}}}{2} \\ &= 1+t+\frac{\epsilon}{2}-\left(\frac{1}{\sqrt{3-2t}}+\frac{1}{\sqrt{3-2(t+\epsilon)}}\right), \end{aligned}$$

except possibly for one case when the interval $[3-2(t+\epsilon), 3-2t]$ contains $\sqrt[3]{4}$, for which $t = 0.101101_2 = 0.703125$. Individual check shows that this gives no trouble. To find the largest error in using $\frac{3-\alpha}{2}-d$ to approximate $\frac{2}{\sqrt{\alpha}}$, it suffices for us to look at

$$\begin{aligned} &1+t-d-\frac{2}{\sqrt{3-2t}} \\ &= -\frac{\epsilon}{2}+\left(\frac{1}{\sqrt{3-2(t+\epsilon)}}-\frac{1}{\sqrt{3-2t}}\right) \\ &= -\frac{\epsilon}{2}\left(1-\frac{4}{(3-2(t+\epsilon))\sqrt{3-2t}+(3-2t)\sqrt{3-2(t+\epsilon)}}\right), \end{aligned}$$

whose absolute value for all possible $0.1_2 \leq t \leq 0.111111_2$ is less than or equal to $\frac{\epsilon}{2}\times 0.9543 \approx 2^{-7.067485}$.

Case (iii): bits of f_1' are all zero.

In this case, $\alpha = 1$. Now if e is odd, then $t = 0.100000_2$. The corresponding d is gotten as in **Case (ii)**. Computation shows the error cannot exceed $0.003502 \approx 2^{-8.1576}$.

If e is even, then $t = 0.000000_2$. The corresponding d is gotten as in **Case (i)**. Computation shows the error cannot exceed $0.00386 \approx 2^{-8.0172}$.

Now we reach the following conclusion: The initial guess gives up to 7 correct bits or more if e is even; while up to 8 correct bits or more if e is odd.

2 Iteration Refinement

Apply Reciproot iteration three times to y and multiply the result by x to get an approximation z that matches $\sqrt{x}$ to about 1 ulp. To be exact, we will have

$$-1.0654\,\mathbf{ulp} \leq z - \sqrt{x} < 1\,\mathbf{ulp}. \tag{3}$$

Set rounding mode to *Round-to-nearest* and sequentially do

$$\begin{aligned} y &:= y(1.5 - 0.5xy^2), & (4)\\ y &:= y((1.5 - 2^{-40}) - 0.5xy^2), & (5)\\ z &:= xy, & (6)\\ z &:= z + 0.5z(1 - zy). & (7) \end{aligned}$$

To analyze the accuracy of y and z after each step, without loss of generality, we assume $1 < x < 4$. ($x = 1$ or 4 can be checked individually.) Then $1 < \sqrt{x} < 2$ and $1 > \frac{1}{\sqrt{x}} > 0.1_2$.

- After initial approximation and before (4): y can be written as

  $$y = \frac{1}{\sqrt{x}} + \epsilon,$$

  where $|\epsilon| < 2^{-8.067485}$ if $1 < x < 2$; $|\epsilon| < 2^{-9}$ if $2 \leq x < 4$.

- After (4) and before (5): y can be written as

  $$\begin{aligned} y &= \frac{1}{\sqrt{x}} - 1.5\epsilon^2\sqrt{x} - 0.5\epsilon^3 x + \epsilon' \\ &\equiv \frac{1}{\sqrt{x}} + \epsilon_1, \end{aligned}$$

  where ϵ' is for rounding errors. Note that if $1 < x < 2$, $1.5\epsilon^2\sqrt{x} + 0.5\epsilon^3 x < 2^{-15.04747}$ and if $2 \leq x < 4$, $1.5\epsilon^2\sqrt{x} + 0.5\epsilon^3 x < 2^{-16.4}$. As rounding error at this stage are negligible unless $\epsilon \sim 2^{-26}$, we conclude that: $|\epsilon_1| < 2^{-14.9}$ if $1 < x < 2$; and $|\epsilon_1| < 2^{-16.4}$ if $2 \leq x < 4$. Further more $\epsilon_1 < 0$ unless it is of order 2^{-52}.

- After (5) and before (6): y can be written as

$$\begin{aligned} y &= \frac{1}{\sqrt{x}} - 2^{-40}\left(\frac{1}{\sqrt{x}} + \epsilon_1\right) - 1.5\epsilon_1^2\sqrt{x} - 0.5\epsilon_1^3 x + \epsilon'' \\ &\equiv \frac{1}{\sqrt{x}} - \epsilon_2, \end{aligned}$$

  where ϵ'' is for rounding errors. Note that if $1 < x < 2$, $1.5\epsilon_1^2\sqrt{x} + 0.5\epsilon_1^3 x < 2^{-29.009967}$ and if $2 \le x < 4$, $1.5\epsilon^2\sqrt{x} + 0.5\epsilon^3 x < 2^{-32.24}$. As rounding error at this stage are negligible, we conclude that: $2^{-41} < \epsilon_2 < 2^{-29.0096}$ if $1 < x < 2$; and $2^{-41} < \epsilon_2 < 2^{-32.23}$ if $2 \le x < 4$.

- After (6) and before (7): $z = fl(xy) = \sqrt{x} - \epsilon_2 x + \epsilon_m$, where $|\epsilon_m| \le 2^{-53}$, i.e., at most $\frac{1}{2}$ **ulp** with respect to 1.

- Computations in (7): $fl(zy) < 1$ and

$$fl(zy) = 1 - 2\epsilon_2\sqrt{x} + \epsilon_2^2 x + \frac{\epsilon_m}{\sqrt{x}} + \epsilon'_m + (\text{neg.}\quad \text{terms}),$$

  where $|\epsilon'_m| \le 2^{-54}$, i.e., at most $\frac{1}{4}$ **ulp** with respect to 1. From now on "(neg. terms)" refers to some negligible terms in comparing with the unit in the last place of a corresponding expression. No rounding error in calculating

$$1 - fl(zy) = 2\epsilon_2\sqrt{x} - \epsilon_2^2 x - \frac{\epsilon_m}{\sqrt{x}} - \epsilon'_m + (\text{neg.}\quad \text{terms}).$$

  Note

$$\begin{aligned} & fl(0.5z(1 - zy)) \\ &= (\sqrt{x} - \epsilon_2 x + \epsilon_m)\left(\epsilon_2\sqrt{x} - \frac{\epsilon_2^2 x}{2} - \frac{\epsilon_m}{2\sqrt{x}} - \frac{\epsilon'_m}{2} + (\text{neg.}\quad \text{terms})\right) \\ &= \epsilon_2 x - 1.5\epsilon_2^2 x\sqrt{x} - \frac{\epsilon_m}{2} - \frac{\epsilon'_m}{2}\sqrt{x} + \epsilon''_m + (\text{neg.}\quad \text{terms}), \end{aligned}$$

  where $|\epsilon''_m| \le 2^{-53}|\epsilon_2 x|$. Now

$$fl(z + 0.5z(1 - zy))$$

$$\begin{aligned}
&= \sqrt{x} - \epsilon_2 x + \epsilon_m + \epsilon_2 x - 1.5\epsilon_2^2 x\sqrt{x} - \frac{\epsilon_m}{2} - \frac{\epsilon'_m}{2}\sqrt{x} \\
&\quad + \epsilon''_m + \epsilon'''_m + (\text{neg.} \quad \text{terms}) \\
&= \sqrt{x} - 1.5\epsilon_2^2 x\sqrt{x} + \frac{\epsilon_m}{2} - \frac{\epsilon'_m}{2}\sqrt{x} + \epsilon'''_m + (\text{neg.} \quad \text{terms}) \\
&= \sqrt{x} + \eta,
\end{aligned}$$

where $|\epsilon'''_m| \leq 2^{-53}$, i.e., at most $\frac{1}{2}$ **ulp** with respect to 1. Note

$$\left|\frac{\epsilon_m}{2} - \frac{\epsilon'_m}{2}\sqrt{x} + \epsilon'''_m\right| \leq \frac{1}{4}\,\mathbf{ulp} + \frac{1}{4}\,\mathbf{ulp} + \frac{1}{2}\,\mathbf{ulp} = 1\,\mathbf{ulp}$$

with respect to 1. On the other hand, $0 > -1.5\epsilon_2^2 x\sqrt{x} \geq -0.0654\,\mathbf{ulp}$ if $1 < x < 2$, and $0 > -1.5\epsilon_2^2 x\sqrt{x} \geq -0.0021\,\mathbf{ulp}$ if $2 \leq x < 4$. Therefore we have

$$-1.0654\,\mathbf{ulp} \leq \eta < 1\,\mathbf{ulp}.$$

We have just proved (3).

3 Final Adjustment

By twiddling the last bit of z it is possible to force z to be correctly rounded according to the prevailing rounding mode as follows. Let r and i be copies of the rounding mode and inexact flag before entering the square root program. Also we use the expression $z\pm\,\mathbf{ulp}$ for the next representable floating numbers (up and down) of z.

- Case RN—*round-to-nearest*: In this case, if $z - \sqrt{x} > \frac{1}{2}\,\mathbf{ulp}$, then do $z = z -\,\mathbf{ulp}$; if $z - \sqrt{x} < -\frac{1}{2}\,\mathbf{ulp}$, then do $z = z +\,\mathbf{ulp}$; otherwise z is correctly rounded already.

  Set rounding mode to *round-toward-zero* which means "**chopped**".

  We write $z = \sqrt{x} + \eta \equiv \sqrt{x} + \frac{1}{2}\,\mathbf{ulp} + \epsilon$ where $-1.564\,\mathbf{ulp} \leq \epsilon < \frac{1}{2}\,\mathbf{ulp}$. Note

$$z(z -\,\mathbf{ulp}) \;=\; z^2 - z \times \mathbf{ulp}$$

$$
\begin{aligned}
&= x + 2\eta\sqrt{x} + \eta^2 - \sqrt{x} \times \mathbf{ulp} - \eta \times \mathbf{ulp} \\
&= x + 2\epsilon\sqrt{x} + \left(\frac{1}{2}\,\mathbf{ulp} + \epsilon\right)\left(\epsilon - \frac{1}{2}\,\mathbf{ulp}\right) \\
&= x + 2\epsilon\sqrt{x} + \epsilon^2 - \frac{1}{4}\,\mathbf{ulp}^2.
\end{aligned}
$$

So $z(z - \mathbf{ulp}) \geq x$ if and only if

$$
\epsilon \geq \frac{\frac{1}{4}\,\mathbf{ulp}^2}{\sqrt{x} + \sqrt{x + \frac{1}{4}\,\mathbf{ulp}^2}} < \frac{\mathbf{ulp}^2}{8\sqrt{x}}. \tag{8}
$$

As computations are supposed to be done under *round-toward-zero*, we have $fl(z(z - \mathbf{ulp})) \geq x$ if and only if (8) holds.

Theorem 1 *There is no IEEE double precision floating point number x such that*

$$
z = \sqrt{x} + \frac{1}{2}\,\mathbf{ulp} + \epsilon
$$

is an IEEE double precision floating point number for some

$$
0 \leq \epsilon < \frac{\mathbf{ulp}^2}{8\sqrt{x}}.
$$

On the other hand, we write $z = \sqrt{x} - \frac{1}{2}\,\mathbf{ulp} - \epsilon$ where $-1.5\,\mathbf{ulp} < \epsilon \leq 0.5654\,\mathbf{ulp}$. Note

$$
\begin{aligned}
z(z + \mathbf{ulp}) &= z^2 + z \times \mathbf{ulp} \\
&= x + 2\eta\sqrt{x} + \eta^2 + \sqrt{x} \times \mathbf{ulp} + \eta \times \mathbf{ulp} \\
&= x - 2\epsilon\sqrt{x} - \left(\frac{1}{2}\,\mathbf{ulp} + \epsilon\right)\left(\frac{1}{2}\,\mathbf{ulp} - \epsilon\right) \\
&= x - 2\epsilon\sqrt{x} + \epsilon^2 - \frac{1}{4}\,\mathbf{ulp}^2.
\end{aligned}
$$

So $z(z + \mathbf{ulp}) \geq x$ if and only if

$$
\epsilon \leq \frac{\frac{1}{4}\,\mathbf{ulp}^2}{\sqrt{x} + \sqrt{x + \frac{1}{4}\,\mathbf{ulp}^2}} < \frac{\mathbf{ulp}^2}{8\sqrt{x}}. \tag{9}
$$

As computations are supposed to be done under *round-toward-zero*, we have $fl(z(z + \mathbf{ulp})) \geq x$ if and only if (9) holds.

Theorem 2 *There is no IEEE double precision floating point number x such that*

$$z = \sqrt{x} - \frac{1}{2}\,\mathbf{ulp} - \epsilon$$

is an IEEE double precision floating point number for some

$$0 \leq \epsilon < \frac{\mathbf{ulp}^2}{8\sqrt{x}}.$$

So the following adjustment

```
case RN:             ... round-to-nearest
    if(x<= z*(z-ulp)...chopped) z = z - ulp; else
    if(x<= z*(z+ulp)...chopped) z = z; else z = z+ulp.
```

will produce a z with

$$|z - \sqrt{x}| < \frac{1}{2}\,\mathbf{ulp}$$

Proofs of the above theorems will be given in the next section at the end of this note.

- Case RZ or Case RM—*round-to-zero or round-to-*$-\infty$: In this case, if $z > \sqrt{x}$, then do $z = z - \mathbf{ulp}$; if $z - \sqrt{x} \leq -\mathbf{ulp}$, then do $z = z + \mathbf{ulp}$; otherwise z is correctly rounded already.

  Reset rounding mode to round-to-$+\infty$.

  Note $z^2 > x$ if and only if $z > \sqrt{x}$, which also holds in floating point operations under RP. $(z + \mathbf{ulp})^2 \leq x$ if and only if $z - \sqrt{x} \leq -\mathbf{ulp}$, which also holds in floating point operations under RP.

  So the following adjustment

```
case RZ:case RM:    ... round-to-zero
                        or round-to-negative infinity
```

```
if(x<z*z ... rounded up) z = z - ulp; else
if(x>=(z+ulp)*(z+ulp) ...rounded up) z = z+ulp.
```

will yield correctly rounded result.

- Case RP—*round-to-*$+\infty$. In this case, if $-1\,\mathbf{ulp} \le z - \sqrt{x} < 0$, then do $z = z + \mathbf{ulp}$; else if $z - \sqrt{x} < -\mathbf{ulp}$, then do $z = z + 2\,\mathbf{ulp}$; otherwise z is correctly rounded already.

  Under rounding mode—*round-to-zero*.

  Note if $(z + \mathbf{ulp})^2 < x$ if and only if $z - \sqrt{x} < -\mathbf{ulp}$, which also holds in floating point operations under RZ. $(z + \mathbf{ulp})^2 < x < z^2$ if and only if $-1\,\mathbf{ulp} \le z - \sqrt{x} < 0$, which also holds in floating point operations under RZ.

  So the following adjustment

```
case RP:            ... round-to-positive infinity
    if(x>(z+ulp)*(z+ulp)...chopped) z = z+2*ulp; else
    if(x>z*z ...chopped) z = z+ulp.
```

  will yield correctly rounded result.

To determine whether z is an exact square root of x, we notice an necessary condition for z to be an exact square root of x is that the training 26 bits of z must be zero. So if the training 26 bits of z is not zero, raise *Inexact* flag; else if e is odd and the 26th bit of z is 1 then z is not exact; else if $z^2 \ne x$ (at this moment $fl(z^2) = z^2$), then z is not exact; otherwise z is exact.

4 Proofs of Theorems 1 and 2

Let us prove Theorem 1 first. Assume to the contrary, there were an IEEE double precision floating point number x as described in the theorem. By scaling x and z properly, we may assume that $1 \le x < 4$. Note

$$\begin{aligned} x &= (z - \frac{1}{2}\,\mathbf{ulp} - \epsilon)^2 \\ &= z^2 - z\,\mathbf{ulp} + \frac{1}{4}\,\mathbf{ulp}^2 - 2\epsilon z + \epsilon\,\mathbf{ulp} + \epsilon^2. \end{aligned} \tag{10}$$

As now, $\mathbf{ulp} = 2^{-52}$, and thus $\frac{1}{4}\,\mathbf{ulp}^2 = 2^{-106}$. It is easy to see that in binary form

$$z^2 - z\,\mathbf{ulp} = a_1 a_0.a_{-1}a_{-2}\cdots a_{-104},$$

where a_j's are either 1 or 0 and a_1, a_0 cannot be 0 at the same time. Therefore

$$z^2 - z\,\mathbf{ulp} + \frac{1}{4}\,\mathbf{ulp}^2 = a_1 a_0.a_{-1}a_{-2}\cdots a_{-104}01,$$

which proves ϵ could not be 0. If, however, $\epsilon > 0$, then

$$0 > -2\epsilon z + \epsilon^2 + \epsilon\,\mathbf{ulp} = \epsilon(-2z + \epsilon + \,\mathbf{ulp}) > -2\epsilon z > -\frac{1}{4}\,\mathbf{ulp}^2.$$

Thus the binary expansion of $z^2 - z\,\mathbf{ulp} + \epsilon\,\mathbf{ulp} + \frac{1}{4}\,\mathbf{ulp} - 2\epsilon z + \epsilon^2$ could not match that of x, contradicting (10). Theorem 1 is proved.

To prove Theorem 2, we apply similar trick as we just did. Suppose we had such an IEEE double precision floating point number x as described in the theorem. Without loss of generality, we may assume $1 \le x < 4$. Note

$$\begin{aligned} x &= (z + \frac{1}{2}\,\mathbf{ulp} + \epsilon)^2 \\ &= z^2 + z\,\mathbf{ulp} + \frac{1}{4}\,\mathbf{ulp}^2 + 2\epsilon z + \epsilon\,\mathbf{ulp} + \epsilon^2 \end{aligned} \tag{11}$$

As now, $\mathbf{ulp} = 2^{-52}$, and thus $\frac{1}{4}\,\mathbf{ulp}^2 = 2^{-106}$. It is easy to see that in binary form

$$z^2 + z\,\mathbf{ulp} = a_1 a_0.a_{-1}a_{-2}\cdots a_{-104},$$

where a_j's are either 1 or 0 and a_1, a_0 cannot be 0 at the same time. Since

$$0 < \frac{1}{4}\,\mathbf{ulp}^2 + 2\epsilon z + \epsilon\,\mathbf{ulp} + \epsilon^2 < 2^{-104},$$

the binary expansion of $z^2 + z\,\mathbf{ulp} + \frac{1}{4}\,\mathbf{ulp}^2 + 2\epsilon z + \epsilon\,\mathbf{ulp} + \epsilon^2$ could not match that of x, contradicting (11). Theorem 2 is proved.

References

[1] W. Kahan and K. C. Ng, SQRT, 1986.