Multiplying a matrix by its inverse should give the identity matrix:

```
A =
   150  -100     0
  -100   150   -50
     0   -50    50
>> Ainv = inv(A)
Ainv =
    0.0200    0.0200    0.0200
    0.0200    0.0300    0.0300
    0.0200    0.0300    0.0500
>> format long
>> test = A * Ainv
test =
   1.000000000000000                   0                   0
  -0.000000000000000   1.000000000000000                   0
                   0                   0   1.000000000000000
>> test == eye(3)
ans =
     0     1     1
     0     1     1
     1     1     1
```

The answer is close, but not quite right…

Some calculator exercises:

Evaluate (1 + 1e-13) -1

The answer should be 1e-13 but is zero….

Evaluate (1e13 + 1) – 1e13

The answer should be 1 but is zero ….

Evaluate (1 + 4e-11 + 8e-11) – 1

The answer should be 1.2e-10 but is zero (Casio fx-991) …..

Evaluate (4e-11 + 8e-11 + 1) – 1

The answer should be 1.2e-10 and is (Casio fx-991) …..

All these issues are a result of the way in which values are represented within computers (and within calculators).

Overflow and Underflow:

X = 9.9999e99 + 9.9999e99

Actual answer too large to be represented (overflow)

X = 1.0000e-99 / 1.000e10 =0

Actual answer too small to be represented (underflow)

Adding large and small values together can be a waste of effort:

X = 1.0000e00
Y = 1.0000e-12

X = X + Y

Actual answer is 1.000000000001
Rounding to 5 digits gives 1.0000

X is left containing 1.0000e00

Adding Y to X has absolutely no effect.

```
>> x = 1
x =
     1
>> y = 1e-20
y =
  1.0000e-020
>> x = x + y
x =
     1
```

The order of addition can be important:

X = 0.0004 + 0.0004 + 0.0004 + 10

Correct answer is 10.0012 (impossible to represent)
Actual answer is 10.001 (as close as we can get)

X = 10 + 0.0004 + 0.0004 + 0.0004

Actual answer is 10.000 (each addition has no effect)

When possible, relatively smaller quantities should be added together before being added to relatively larger ones.

Principle applies to both addition and subtraction and to both positive and negative values ("smaller" means smaller magnitude).

The formula used can be important:

Standard Quadratic formula:

$$ax^2 + bx + c = 0$$

$$x_1 = \frac{-b + r}{2a} \quad x_2 = \frac{-b - r}{2a} \quad \text{where} \quad r = \sqrt{b^2 - 4ac}$$

Rationalized Quadratic Formula (same thing re-arranged algebraically):

$$x_1 = \frac{2ac}{-b - r} \quad x_2 = \frac{2ac}{-b + r} \quad \text{where} \quad r = \sqrt{b^2 - 4ac}$$

For $x^2 - 97x + 1 = 0$

	x_1	x_2
Exact	96.9896896	0.0103103743
SQF (5 digits)	96.990	0.01050
RQF (5 digits)	95.238	0.01031

Formulas involving $(-b - r)$ give a poor result. It is best to avoid subtracting quantities that are close to each other ("cancellation error"). Note: $-b = 97$, $r = 96.979$

Example of Numerical Issues (ECOR 1606 Test Problem)

The shortest distance between a straight line connecting two points (P_1 and P_2) and a third point (P_3) can be calculated as follows:

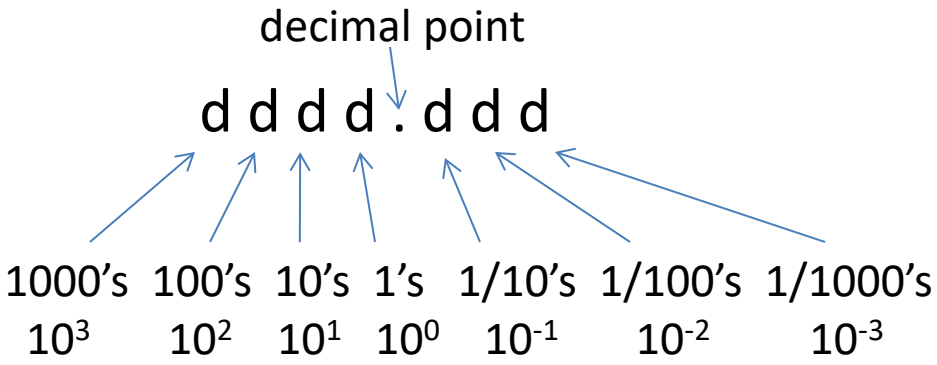
let D be the distance we're looking for
D_{13} be the distance between P_1 and P_3,
D_{23} be the distance between $P_2$and P_3,
D_{12} be the distance between $P_1$and P_2, and

if $D_{13}^2 >= D_{23}^2 + D_{12}^2$: $D = D_{23}$
if $D_{23}^2 >= D_{13}^2 + D_{12}^2$: $D = D_{13}$
otherwise:

$$D = \sqrt{D_{23}^2 - \left(\frac{D_{12}^2 + D_{23}^2 - D_{13}^2}{2D_{12}} \right)^2}$$

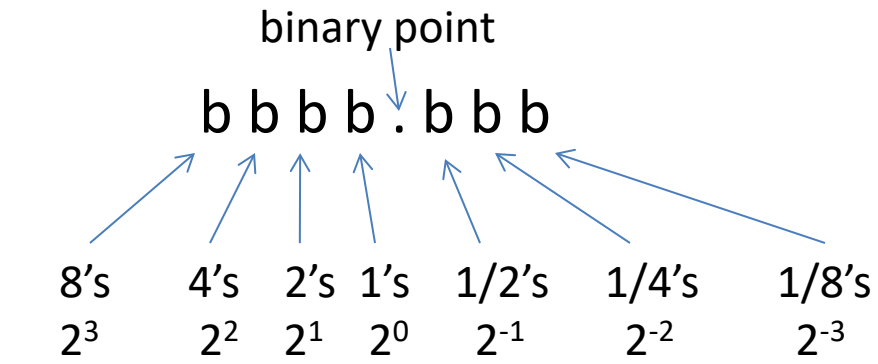
In theory the quantity under the square root sign can never be negative. In practice numerical errors can result in it being slightly negative (in which case taking the square root causes problems).

Decimal (base 10)



10 possible digits (0, 1, 2, 3, 4, … , 9)

Binary (base 2)



Just 2 possible digits (0, 1)

Counting in Binary:

$0_{10} = 0_2$
$1_{10} = 1_2$
$2_{10} = 10_2$
$3_{10} = 11_2$
$4_{10} = 100_2$
$5_{10} = 101_2$
$6_{10} = 110_2$
$7_{10} = 111_2$
$8_{10} = 1000_2$
$9_{10} = 1001_2$
$10_{10} = 1010_2$
$11_{10} = 1011_2$
…

Any integer value can be represented in binary. It just takes a lot more digits than are required in decimal.

Conversion Examples:

$110.11_2 = (1 \times 4) + (1 \times 2) + (1 \times ½) + (1 * ¼) = 6.75$

$10.001_2 = (1 \times 2) + (1 \times 1/8) = 2.125$

Scientific Notation:

Scientific notation in binary is analogous to the decimal version. The only difference is that it involves powers of 2 rather than 10. The exponent indicates how many positions the binary point should be shifted.

$110.11_2 \times 2^6 = 110110000_2$

$1.1011_2 \times 2^{-5} = 0.000011011_2$

Significant Digits

53 significant binary digits corresponds to about 15 to 16 significant decimal digits

$2^{10} = 1024 \approx 10^3 = 1000$

Therefore 10 binary digits correspond to about 3 decimal digits (each binary equates to about 0.3 decimal digits).

0.3 * 53 = 15.9

Overflow and Underflow:

```
>> format compact          Avoid double spacing
>> x = 1e306
x =
  1.0000e+306
>> x = x * 1e50
x =
   Inf                    Overflow (result too large)
>> x = 1e-306
x =
  1.0000e-306
>> x = x / 1e50
x =
     0                    Underflow (result too small)
>> (1e-306 / 1e50) * 1e50
ans =
     0
```

Implications of Binary Representation:

- Number of significant decimal digits somewhat ambiguous

- Conversion between decimal and binary introduces further errors

Example: Something as basic as the fraction $1/10 = 0.1_{10}$ presents problems because it can not be precisely represented in binary.

$$1/10 = 0.1_{10} = 0.0001100110011001100110011\ldots._2$$

```
>> s = 0;
>> for k = 1 : 10000
        s = s + 0.0001;   % 0.0001 cannot be represented exactly
    end
>> format long
>> s
s =
  0.99999999999906    % the very small error adds up. Not
with1/(2^20) added together 2^20 times
```