3.4 Kahan's Machine Epsilon

These statements were first given by Prof William Kahan, Berkeley.

```
a = 4/3;  b = a-1;  c = b+b+b;  e = 1-c;
```

Mathematically we have

$$a = \frac{4}{3}, \quad b = \frac{4}{3} - 1 = \frac{1}{3}, \quad c = \frac{1}{3} + \frac{1}{3} + \frac{1}{3} = 1, \quad e = 1 - 1 = 0.$$

Performing these statements in $F(b, p, -, -)$, where b is not a multiple (power?) of 3, we get

1. $a = \text{fl}(4/3) \;= \text{fl}(1.33\ldots33\ldots) \;= \underbrace{1.33\ldots3}_{p \text{ digits}}$
2. $b = \text{fl}(\text{fl}(a) \;- 1) \;= \text{fl}(1.33\ldots3 - 1.0) \;= 0.\underbrace{33\ldots3}_{p-1}$
3. $c = \text{fl}(b + b + b) \;= \text{fl}(0.33\ldots3 + 0.33\ldots3 + 0.33\ldots3) \;= 0.\underbrace{99\ldots9}_{p-1}$
4. $e = \text{fl}(1 - c) \;= \text{fl}(1.0 - 0.\underbrace{99\ldots9}_{p-1}) \;= 0.\underbrace{00\ldots1}_{p-1} = 1.00\ldots0 \times b^{1-p}$

Thus we get $e = b^{1-p} = \epsilon_m$. We have implicitly assumed that $b = 10$.

Notice that the only rounding error occurs in the statement `a = 4/3`. This rational number does not have a finite expansion in base 2 or 10 and so there will always be a rounding error, no matter how large p is.

Here is a small Fortran function that uses Kahan's ϵ_m calculation. This function was in the Eispack subroutine package, and is still part of Dongarra's Linpack benchmark program `1000d.for`. Note the starred sentence at the end of the comments.

```
C=======================================================================
      double precision function epslon (x)
C=======================================================================
      double precision x
c
c     estimate unit roundoff in quantities of size x.
c
      double precision a,b,c,eps
c
c     this program should function properly on all systems
c     satisfying the following two assumptions,
c        1.    the base used in representing dfloating point
c              numbers is not a power of three.
c        2.    the quantity  a  in statement 10 is represented to
c              the accuracy used in dfloating point variables
c              that are stored in memory.
c     the statement number 10 and the go to 10 are intended to
c     force optimizing compilers to generate code satisfying
c     assumption 2.
c     under these assumptions, it should be true that,
c              a  is not exactly equal to four-thirds,
c              b  has a zero for its last bit or digit,
c              c  is not exactly equal to one,
c              eps  measures the separation of 1.0 from
c                   the next larger dfloating point number.
c     the developers of eispack would appreciate being informed          *
c     about any systems where these assumptions do not hold.             *
c
c     ******************************************************************
c     this routine is one of the auxiliary routines used by eispack iii
c     to avoid machine dependencies.
c     ******************************************************************
c
c     this version dated 4/6/83.
c
      a = 4.0d0/3.0d0
   10 b = a - 1.0d0
      c = b + b + b
      eps = dabs(c-1.0d0)
      if (eps .eq.  0.0d0) go to 10
      epslon = eps*dabs(x)
      return
      end
```

3.5 Calculations with π and pi

We know that $\sin(\pi) = 0$, $\cos(\pi) = -1$, and $\sin^2(\pi) + \cos^2(\pi) = 1$. But MATLAB gives

1. `sin(pi)` $= 1.224646799147353\,e - 016$
2. `cos(pi)` $= -1$
3. `sin(pi)^2+cos(pi)^2` $= 1$

As we can see, 2 and 3 are correct but 1 is not. There are two sources of error here : (i) the error in representing π, a transcendental number, and (ii) the error in computing $\sin(x)$ or any other function.