Roundoff error in Euler's Method - An Example

In Turbo Pascal real numbers have 11 significant digits. In ANSI C the number of significant digits that a single or double precision floating point number can store is specified by FLT_DIG and DBL_DIG, which are defined in <float.h>. For example on SPARC's, a single precision number has 6 significant decimal digits while a double has 15. Suppose we perform a computation of N steps in which each step independently introduces a random roundoff error whose mean is zero (i.e. it is negative as often as it is positive) and whose magnitude has mean of order 10^{-15}. Then by the central limit theorem, the magnitude of the total roundoff error introduced will be, on average, about $\sqrt{N}10^{-15}$.

So as a general rule roundoff error isn't too much of a problem, but there is one standard situation when it is a very big problem ideed. Catastophic cancelations can occur when we compute the difference $a - b$ of two very large, but close together numbers a, b. It is possible for the roundoff error in a and b to be much larger than the real value of the difference $a - b$, resulting in a huge percentage error in the computed value of $a - b$. For example

$$(10,000,000,001 \pm 1) - (10,000,000,000 \pm 1) = 1 \pm 2$$

The following table is designed to illustrate the effects of roundoff error when Euler's method is used to generate an approximate solution to the initial value problem

$$y' = f(t, y)$$
$$y(0) = y_0$$

n	y_n	Y_n	$\phi - y_n$	$y_n - Y_n$	$\frac{y_n - Y_n}{\sqrt{n}}$
10	34.411	34.250	30.486	0.161	0.051
20	45.588	46.500	19.309	-0.912	-0.204
40	53.808	54.000	11.090	-0.192	-0.030
80	58.917	59.250	5.981	-0.333	-0.037
160	61.786	62.500	3.112	-0.714	-0.056
320	63.310	63.750	1.588	-0.440	-0.025
640	64.096	47.500	0.802	16.596	0.656
1280	64.495	2.000	0.403	62.495	1.747
2560	64.696	1.000	0.202	63.696	1.259
5120	64.797	1.000	0.101	63.797	0.892

In the table

$$
\begin{aligned}
f(t,y) &= 1 - t + 4y \\
y_0 &= 1 \\
n &= \text{number of steps} \\
y_n &= \text{approximate value for } y(1) \text{ with full double precision} \\
Y_n &= \text{approximate value for } y(1) \text{ with all operations rounded to 8 binary digits} \\
\phi &= \text{exact value for } y(1)
\end{aligned}
$$

The computations were done using a compiler for which "double precision" means fifteen significant decimal digits. So the five digits of y_n in the table are free of roundoff error. For the last rows of the table, h is so small that the $hf(t_j, y_j)$ term in $y_{j+1} = y_j + hf(t_j, y_j)$ is always rounded to exactly zero.