Sample Homework Solutions
Chapter 1: Scientific Computing

Yu-Rong Wang and Cheng-Hsin Hsu

The solutions are for your reference only. If you have any doubts or thoughts about the sample answers, please let the instructor and the TA know. More importantly, like other math questions, the homework questions may be solved in multiple ways; potentially because of different assumptions (made by you, the students). Do not assume the sample solutions here are the only *correct* answers. Please discuss with others about alternative solutions.

We will not grade your homework assignments. You are, however, highly encouraged to discuss with us during the Lab session and office hours. The correlation between the homework assignments and quiz/midterm/final questions is high. So you do want to practice more and sooner.

1 Review Questions

- **1.5** True. The propagated data errors come from the imperfect inputs, rather than the algorithms.
- **1.10** False. The underflow level is the smallest positive number that can be represented by the floating number system.
- **1.15** Absolute error shows the absolute difference between the approximate value and the true value. It may happen that the absolute error is large but has little effect on the results. Relative error shows the proportion of absolute error to true value. If the relative error is larger, the approximation is worse.
- **1.20** (Relative) condition number is the ratio of relative change in the solution to the relative change in the input. It is usually the most appropriate measure for the sensitivity of a problem. However, it is not defined if either the input or output is zero. In this case, the absolute condition number (defined by the ratio of changes without normalization) is an appropriate measure of sensitivity.
- **1.25** See Example 1.9 and Figure 1.3. The distribution is not uniform along the real line. It is denser when being closer to zero, and becomes sparser when being farther from zero.

- **1.30** The unit roundoff ϵ_{mach} determines the maximum possible relative error in representing a given nonzero real number in a floating-point system. More specifically, we know:

$$\left|\frac{\text{fl}(x) - x}{x}\right| \leq \epsilon_{\text{mach}}. \tag{1}$$

ϵ_{mach}, also known as machine precision and machine epsilon is the smallest number such that:

$$\text{fl}(1 + \epsilon) > 1. \tag{2}$$

- **1.35** No. A counter example: let's divide $(1.00)_2 \times 2^{-1} = (0.5)_{10}$ by $(1.10)_2 \times 2^{-1} = (0.75)_{10}$ in a floating-point system with $\beta = 2$, $p = 3$, L = -1, $U = 1$ and rounding by chopping. The result of the floating-point division is $(1.01)_2 \times 2^{-1} = (0.625)_{10}$, which is different from the real arithmetic result of 2/3.

- **1.40** The relative error in representing any nonzero number x is less than or equal to unit roundoff $\epsilon_{mach} = \frac{1}{2}\beta^{1-p}$.

- **1.45** The sequence should sum from the smallest number to the largest number, which can minimize rounding error.

- **1.50** One of the factors is normalization because normalization will lead in a gap between the underflow level UFL and zero. Another factor is larger exponents, the steps (granularity) are wider (coarser).

2 Exercises

- **1.6**

(a)

forward error:

x	$\hat{y}$	y	Δy
0.1	0,1	$\sin(0.1) \approx 0.09983342$	0.00016658
0.5	0.5	$\sin(0.5) \approx 0.47942554$	0.02057446
1.0	1.0	$\sin(1.0) \approx 0.84147098$	0.15852902

backward error:

$\hat{y}$	$\hat{x} = \arcsin(\hat{y})$	Δx
0.1	$\arcsin(0.1) \approx 0.10016742$	0.00016742
0.5	$\arcsin(0.5) \approx 0.52359878$	0.02359878
1.0	$\arcsin(1.0) \approx 1.57079633$	0.57079633

(b)

forward error:

x	$\hat{y}$	y	Δy
0.1	0.09983333	$\sin(0.1) \approx 0.09983342$	-0.00000009
0.5	0.47916667	$\sin(0.5) \approx 0.47942554$	-0.00025887
1.0	0.83333333	$\sin(1.0) \approx 0.84147098$	-0.00813765

backward error:

$\hat{y}$	$\hat{x} = \arcsin(\hat{y})$	Δx
0.1	$\arcsin(0.09983333) \approx 0.09999991$	-0.00000009
0.5	$\arcsin(0.47916667) \approx 0.49970505$	-0.00029495
1.0	$\arcsin(0.83333333) \approx 0.98511078$	-0.01488922

- **1.12** Almost in all cases, $(x-y)(x+y)$ is more accurate, because $x^2 - y^2$ suffers from catastrophic cancellation. Here, cancellation is due to potentially large rounding errors of x^2 and y^2, especially when x and y can be exactly represented. Now, check $(x-y)$ and $(x+y)$, their rounding error is at most ϵ_{mach}. Multiplying them together leads to even smaller rounding error of ϵ_{mach}^2.
  Now, let's consider a real example: assume a 2-digit system with $x = 8.9 \times 10^2$ and $y = 8.8 \times 10^2$. We have:

$$x^2 - y^2 = 7.9 \times 10^5 - 7.7 \times 10^5 = 2 \times 10^4, \tag{3}$$

  and

$$(x+y)(x-y) = (17 \times 10^2)(1 \times 10) = 1.7 \times 10^4. \tag{4}$$

  The actual answer is 1.77×10^4. Therefore $(x+y)(x-y)$ is more accurate in this example.
  Exercise: can you find an example where $x^2 - y^2$ is more accurate?

- **1.18** There are $2 \times 2^{23} \times (127 - 126 + 1) = 127 \times 2^{25}$ normalized machine numbers in a single-precision IEEE floating-point system. If subnormals are allowed, we will have $2 \times (2^{23} - 1) = 2^{24} - 2$ additional machine numbers.

- **1.24** Without loss of generality, we assume x and y are both positive and $1 \le x/y \le \beta$. We can do this, because if $\beta \le x/y < 1$, we just relabel x as y, and y as x. Let $p \in \mathbb{N}$ be the precision, $\beta = 2$ be the base. We write:

$$x = \big(d_0 + \frac{d_1}{2} + \frac{d_2}{2^2} + \cdots + \frac{d_{p-1}}{2^{p-1}}\big)2^e, \tag{5}$$

  and

$$y = \big(d_0' + \frac{d_1'}{2} + \frac{d_2'}{2^2} + \cdots + \frac{d_{p-1}'}{2^{p-1}}\big)2^{e'}. \tag{6}$$

  Since $1 \le x/y \le 2$, we know $e = e'$ or $e = e' + 1$. For the former case, we have:

$$x - y = \big((d_0 - d_0') + \frac{d_1 - d_1'}{2} + \frac{d_2 - d_2'}{2^2} + \cdots + \frac{d_{p-1} - d_{p-1}'}{2^{p-1}}\big)2^e, \tag{7}$$

  which is exact as $d_i, d_i' \in \{0, 1\}, \forall i$.

  Now, consider the case where $e = e' + 1$. $x/y < 2$ actually means: *if we shift* $d_0' + d_1', \ldots, d_{p-1}'$ *to the right for 1 digit*, compare each digit d_i' (after being shifted)

against d_i from left to right, we will be able to find at least one digit $d'_i > d_i$; actually, because $\beta = 2$, we know $d'_i = 1$ and $d_i = 0$. Now, we pick the smallest i that have this property, and denote it as t, where $0 \leq t \leq p-1$. We will see in a second, why we need t

Coming back to our problem, and let's align the digits of x and y:

$$\begin{array}{cccccccccc} & d_0 & d_1 & \dots & \dots & d_t & \dots & \dots & d_{p-1} & \\ - & & d'_0 & d'_1 & \dots & d'_{t-1} & d'_t & \dots & d'_{p-2} & d'_{p-1} \\ \hline & \hat{d}_0 & \hat{d}_1 & \dots & \dots & \hat{d}_t & \dots & \dots & \hat{d}_{p-1} & \hat{d}_p \end{array} \tag{8}$$

Our goal is to show that $\hat{d}_0 = 0$, so that $x - y$ can be represented with p binary digits. If $d_0 = 0$, we immediately have $\hat{d}_0 = 0$; so we only need so consider the case $d_0 = 1$. Now we scan from d_1, d_2, and up to d_t. We stop at $d_u = 0$ (first zero, i.e., $d_i = 1 \forall 0 \leq i < u$). We note that u exists, because the worst case is $u = t$. Now, observe that the subtraction of the first u digits:

$$\begin{array}{cccccccc} & d_0 & d_1 & \dots & \dots & d_u & \dots & \dots \\ - & & d'_0 & d'_1 & \dots & d'_{u-1} & d'_u & \dots \\ \hline \end{array} \tag{9}$$

can be written as:

$$\begin{array}{cccccccc} & 1 & 1 & \dots & \dots & 1 & \boxed{0} & \dots \\ - & & 1 & 1 & \dots & 1 & 1 & \dots \\ \hline & \boxed{0} & 1 & \dots & \dots & 1 & 1 & \dots \end{array} \tag{10}$$

Notice that $\hat{d}_0 = 0$, because we borrow from d_{u-1}, d_{u-2}, $\dots$, d_1, and d_0. This yields our proof with $\beta = 2$.

When $\beta \geq 3$, it should not be hard to see that the *cascading borrows* happened with $\beta = 2$ may not happen. Let $\beta = 3$ and $p = 3$. If we write $x = (121)_3 \times 3^1$ and $y = (122)_3 \times 3^0$. We have:

$$x - y = (1210)_3 - (122)_3 = (1011)_3, \tag{11}$$

which can not be exactly represented with $p = 3$.