Floating-point arithmetic and error analysis (AFAE)

Final exam - 4 February 2025

Documents allowed. Calculators allowed - Duration: 2h00

The points are given as an indication

The subject is divided into 8 independent exercises. The quality of the writing, the clarity and the precision of the reasoning will play an important part in the evaluation.

Exercise 1 (Floating-point arithmetic (4 points)). Let x be a floating-point number in IEEE 754 double precision such that $1 \leq x < 2$. Show that, when rounding to the nearest, $\mathrm{fl}(x \times \mathrm{fl}(1/x))$ is either 1 or $1 - \mathbf{u}$, where $\mathbf{u} = 2^{-53}$.

Exercise 2 (Absorption (4 points)). Let M be a floating-point number large enough such that $\mathrm{fl}(10 + M) = M$. What are the possible values of $\mathrm{fl}(\sum_{i=1}^{6} x_i)$ where $\{x_i\}_{i=1}^{6} = \{1, 2, 3, 4, M, -M\}$, assuming the sum is evaluated using the classical recursive summation algorithm?

Exercise 3 (FastTwoSum with directed rounding (4 points)). We assume we are working in IEEE 754 double precision and have two double precision floating-point numbers a and b **such that** $|a| \geq |b|$. The `FastTwoSum` algorithm is recalled below.

Algorithm 1 FastTwoSum

function $[s, t] =$ `FastTwoSum`(a, b)

1: $s \leftarrow \mathrm{fl}(a + b)$
2: $z \leftarrow \mathrm{fl}(s - a)$
3: $t \leftarrow \mathrm{fl}(b - z)$

If rounding to the nearest is used, we have $s + t = a + b$, where t is the rounding error, which can be represented as a double-precision floating-point number.

1. Show that the rounding error is no longer necessarily representable if rounding towards $+\infty$ is used. To do so, provide a counterexample.
2. Now suppose we are working with rounding towards $+\infty$. Thus, $s + e = a + b$, where e is a real number that is not necessarily a floating-point number. Show that in the `FastTwoSum` algorithm, we indeed have $z = s - a$. To do this, use Sterbenz's lemma, distinguishing the cases $a, b \geq 0$ and $a \geq 0, b \leq 0$ (in this case, you may further distinguish $-b \geq a/2$ and $-b < a/2$). You do not need to handle the cases $a, b \leq 0$ and $a \leq 0, b \geq 0$.
3. Deduce that $|t - e| \leq 2\mathbf{u}|e|$, where $\mathbf{u}$ is the unit roundoff ($\mathbf{u} = 2^{-53}$ in double precision).

Exercise 4 (Stochastic arithmetic (7 points)).

We consider the linear system $AX = B$ with $A = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$, $B = \begin{pmatrix} 0 \\ p \end{pmatrix}$, $a = b + 1$ and $c = b - 1$.

As a remark, this system is equivalent to $A'X = B$ with $A' = \begin{pmatrix} 1 & b/a \\ 0 & c - b^2/a \end{pmatrix}$.

Please turn the page

1. Show that its exact solution is $X_{sol} = \begin{pmatrix} pb \\ -pa \end{pmatrix}$.

2. Let $b = 303$ and $p = 3$. The linear system is solved using Gaussian elimination, then the computed solution, and the solution obtained from X_{sol} are displayed.

   With IEEE binary32 one obtains

   - with classic floating-point arithmetic, using rounding to nearest:

     ```
     X(0)= 9.07950562e+02 (exact solution: Xsol(0)= 9.09000000e+02)
     X(1)=-9.10947083e+02 (exact solution: Xsol(1)=-9.12000000e+02)
     ```

   - with DSA (Discrete Stochastic Arithmetic) implemented in CADNA (in this case, only digits estimated correct are displayed):

     ```
     X(0)= 0.9E+003 (exact solution: Xsol(0)= 0.9089999E+003)
     X(1)=-0.9E+003 (exact solution: Xsol(1)=-0.9120000E+003)
     There is  1 numerical instability
     1 LOSS(ES) OF ACCURACY DUE TO CANCELLATION(S)
     ```

   For both executions, compare the computed results and those obtained from X_{sol}.

   Compare the results computed with and without DSA.

   Are the results consistent? Justify your answer.

3. Describe the kind of instability detected in DSA. In which situation does it happen?

   Which computation is responsible for this instability? Why?

4. May this kind of instability invalidate the estimation of accuracy by DSA? What are the different kinds of instabilities that may invalidate the estimation of accuracy by DSA?

5. With IEEE binary64 one obtains

   - with classic floating point arithmetic, using rounding to nearest:

     ```
     X(0)= 9.090000000078280e+02 (exact solution: Xsol(0)= 9.090000000000000e+02)
     X(1)=-9.120000000078539e+02 (exact solution: Xsol(1)=-9.120000000000000e+02)
     ```

   - with DSA:

     ```
     X(0)= 0.9089999999E+003 (exact solution: Xsol(0)= 0.908999999999999E+003)
     X(1)=-0.9119999999E+003 (exact solution: Xsol(1)=-0.912000000000000E+003)
     There is  1 numerical instability
     1 LOSS(ES) OF ACCURACY DUE TO CANCELLATION(S)
     ```

   Again, for both executions, compare the computed results and those obtained from X_{sol}.

   Compare the results computed with and without DSA.

   Are the results consistent? Justify your answer.

6. Is the accuracy in binary64 consistent with the one in binary32? Justify your answer.

7. Changing b to 3143756 one obtains with DSA in binary64:

   ```
   X(0)= 0.94E+007 (exact solution: Xsol(0)= 0.943126799999999E+007)
   X(1)=-0.94E+007 (exact solution: Xsol(1)=-0.943127100000000E+007)
   There is  1 numerical instability
   1 LOSS(ES) OF ACCURACY DUE TO CANCELLATION(S)
   ```

How do you explain this change of accuracy?

What should be the accuracy in binary32? Justify your answer.

Exercise 5 (Fixed-point arithmetic (6 points))**.**

1. Consider the signed fixed-point format $(6, -4)$ (most significant bit position: 6, least significant bit position: -4). Indicate the number of bits in this format, and the smallest and largest values. What is the quantization step?

2. Represent the following reals in 8-bit signed fixed point, indicating the fixed point format (position of the most significant bit and the least significant bit).

$$9.3452 \qquad 0.03434 \qquad -2.18625$$

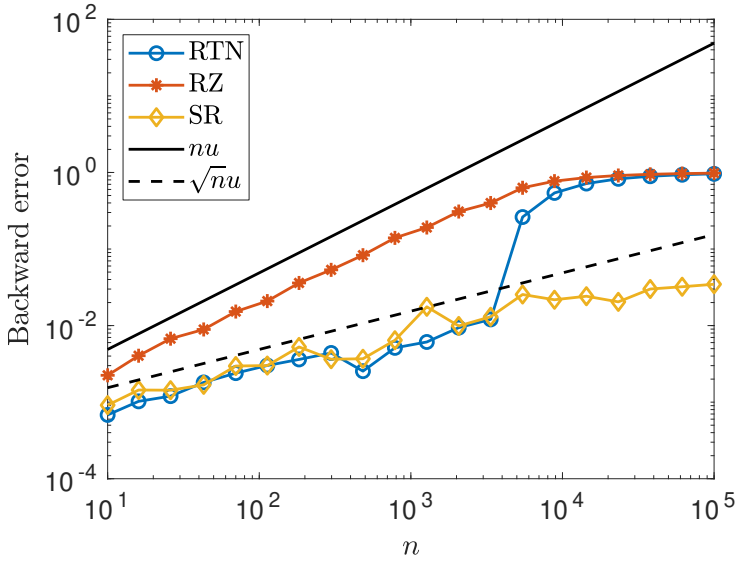
3. Indicate the relative and absolute error for each. In a general way, how to bound these two errors as a function of the width w used to represent the number.

4. We want to calculate

$$9.3452x + 0.03434y - 2.18625z$$

where x, y and z are variables known to belong to the intervals $[-24; 3.4]$, $[-150; 2500]$ and $[-112; 18]$ respectively. Propose a fixed-point format for these three variables, as well as a fixed-point realization (set of mantissa operations) to best perform this operation, assuming we have an 8-bit by 8-bit multiplier giving a 16-bit result, and a 16-bit accumulator (plus 4 guard bits).

Exercise 6 (Rounding error accumulation (2 points))**.** Analyze and discuss the figure below, which compares the backward error of the matrix–vector product $y = Ax$ computed in fp16 arithmetic, where the coefficients of $A \in \mathbb{R}^{n \times n}$ and x have been randomly generated in $[0, 1]$, for three different rounding modes: round-to-nearest (RTN), round-to-zero (RZ), and stochastic rounding (SR).



Exercise 7 (Mixed precision blocked summation (3 points))**.** We consider the summation algorithm below, which is parameterized by three precisions u_1, u_2, and u_3, and by two block sizes b_1 and b_2 (we assume for simplicity that n is a multiple of $b_1 b_2$).

Please turn the page

Input: $x_1, \dots, x_n$
Output: $s = \sum_{i=1}^{n} x_i$
for $i = 1$ **to** $\frac{n}{b_1}$ **do**
  $y_i = \sum_{j=(i-1)b_1+1}^{ib_1} x_j$ in precision u_1
end for
for $i = 1$ **to** $\frac{n}{b_1 b_2}$ **do**
  $z_i = \sum_{j=(i-1)b_2+1}^{ib_2} y_j$ in precision u_2
end for
$s = \sum_{j=1}^{\frac{n}{b_1 b_2}} z_j$ in precision u_3

1. Draw the summation tree corresponding to this algorithm for $n = 12$, $b_1 = 3$, and $b_2 = 2$ (as a reminder, in a summation tree the leaves correspond to the summands x_i, the root corresponds to the sum s, and the rest of the nodes correspond to partial sums). You will assign to non-leaf nodes the labels 1, 2, or 3 to indicate which precision u_1, u_2, or u_3 is used.

2. The computed sum $\widehat{s}$ satisfies a backward error bound of the form

$$\widehat{s} = \sum_{i=1}^{n} x_i(1+\theta_i), \quad |\theta_i| \le f(n, b_1, b_2, u_1, u_2, u_3).$$

   What is the expression of $f(n, b_1, b_2, u_1, u_2, u_3)$?

3. If $u_1 = u_2 = u_3$, which choice of b_1 and b_2 minimizes the bound?

4. If we want to use three different arithmetics (such as fp64, fp32, and fp16), which arithmetic do you propose to assign to which parameter u_1, u_2, u_3? Why?

Exercise 8 (Mixed precision iterative refinement (2 points))**.** The following iterative refinement is applied to a linear system $Ax = b$ with $\kappa(A) = 10^3$.

1: Compute the factorization $A = LU$ in precision u_f.
2: Compute $x = U^{-1}L^{-1}b$ in precision u_f.
3: **for** $i = 1$ **to** n_{iter} **do**
4: Compute the residual $r = b - Ax$ in precision u_r.
5: Compute $d = U^{-1}L^{-1}r$ in precision u_f.
6: Compute $x = x + d$ in precision u.
7: **end for**

With $u_f = u = u_r =$ fp32 and $n_{\text{iter}} = 1$, the algorithm produces a computed solution $\widehat{x}$ achieving a forward error

$$\varepsilon_{\text{fwd}} = \frac{\|\widehat{x} - x\|}{\|x\|} \approx 10^{-4},$$

as indicated in the table below.
Finish completing the table with the order of magnitude of the values that ε_{fwd} would take if we change the parameters n_{iter}, u_f, u or u_r of the algorithm as indicated. No need to justify your answers. As a reminder, the unit roundoffs of fp64, fp32, fp16, and bfloat16 arithmetics are $2^{-53} \approx 10^{-16}$, $2^{-24} \approx 10^{-7}$, $2^{-11} \approx 10^{-4}$ and $2^{-8} \approx 10^{-3}$, respectively.

	$n_{\text{iter}} = 1$	$n_{\text{iter}} = 10$
$u_f = u = u_r = \text{fp32}$	10^{-4}	
$u_f = u = \text{fp32}$, $u_r = \text{fp64}$		
$u_f = u_r = \text{fp32}$, $u = \text{fp64}$		
$u_f = \text{fp32}$, $u = u_r = \text{fp64}$		
$u = u_r = \text{fp32}$, $u_f = \text{fp64}$		
$u = u_r = \text{fp64}$, $u_f = \text{fp16}$		
$u = u_r = \text{fp64}$, $u_f = \text{bfloat16}$		