Exercise Sheet 1

IEEE-754 Floating Point Arithmetic & Linear Systems

1. IEEE-754 Properties

Answer which of the following statements unconditionally hold in IEEE-754 arithmetic, assuming that the variables a, b, c, d can take on *any finite* value.

True	False	
☐	☐	$(a+b)+(c+d)=(d+c)+(b+a)$.
☐	☐	$-a+a+a=a$.
☐	☐	$a \cdot a \geq 0$.
☐	☐	$(a \cdot b) \cdot c = a \cdot (b \cdot c)$
☐	☐	Denormalized (subnormal) numbers should be avoided in calculations as they cause numerical errors.
☐	☐	If $a < b$, then $a + c < b + c$.

Solution:

1. **True**: Floating point addition is commutative, so $(a+b)+(c+d)=(d+c)+(b+a)$ holds.
2. **True**: $-a+a+a=a$ evaluates left-to-right: $(-a+a)+a=0+a=a$. The subtraction $-a+a$ produces zero for finite values.
3. **True**: $a \cdot a \geq 0$ is always true for finite values. The square of any real number (including negative) is non-negative, and this holds in IEEE-754.
4. **False**: Floating point multiplication is not associative. Let $a = 10^{308}$, $b = 2$, $c = 0.5 \times 10^{-308}$. Then $(a \cdot b) \cdot c$ overflows to infinity first, while $a \cdot (b \cdot c) = a \cdot 10^{-308} = 1$.
5. **False**: Au contraire, denormalized numbers exist to *reduce* errors close to the underflow-to-zero condition. However, they can cause severe performance issues on some hardware.
6. **False**: While this seems like it should be true, rounding errors can break this monotonicity property. For example, if a and b are very close and c is much larger in magnitude, the additions $a + c$ and $b + c$ might round to the same value, making them equal rather than maintaining the strict inequality.

2. Fondue Recipe

You're hosting a fondue party and want to create the perfect cheese blend. Traditional Swiss fondue uses a mix of cheeses, and you have access to three types:

- **Vacherin Fribourgeois**: 30% fat content, CHF 2.00 per 100g
- **Gruyère**: 32.5% fat content, CHF 2.10 per 100g
- **Appenzeller**: 31% fat content, CHF 2.40 per 100g

Your recipe requires:

- Exactly 600g of cheese total

- An average fat content of 31%
- A total cost of CHF 12.60

Let v, g, and a be the amounts (in grams) of Vacherin, Gruyère, and Appenzeller, respectively.

a) Explain why finding the perfect blend is equivalent to solving a linear system.

b) Write down the 3×3 matrix A and vector $\mathbf{b}$ such that $A\mathbf{x} = \mathbf{b}$, where $\mathbf{x} = (v, g, a)^T$.

Solution:

a) Each requirement gives us one linear equation in the unknowns v, g, and a: - Total weight: $v + g + a = 600$ - Total fat: $0.30v + 0.325g + 0.31a = 186$ (since 31% of 600g = 186g of fat) - Total cost: $0.02v + 0.021g + 0.024a = 12.60$ (converting to CHF per gram)

These are all linear in the unknowns, giving us a 3×3 linear system.

b) The system is:

$$\begin{pmatrix} 1 & 1 & 1 \\ 0.30 & 0.325 & 0.31 \\ 0.02 & 0.021 & 0.024 \end{pmatrix} \begin{pmatrix} v \\ g \\ a \end{pmatrix} = \begin{pmatrix} 600 \\ 186 \\ 12.60 \end{pmatrix}$$

3. Exponential Function

The mathematical utility library of most programming languages (including NumPy) comes with a function named `expm1`, which computes

$$\texttt{expm1}(x) \coloneqq e^x - 1$$

in IEEE-754 floating point arithmetic. The manual of this function advises developers to

"replace expressions of the form $\texttt{exp}(x) - 1$ with $\texttt{expm1}(x)$."

What could be the purpose of such a specialized function? Is the advice always valid, or can we ignore it in some cases?

Solution:

For small values of x, $\exp(x) \approx 1$ and the subtraction suffers from severe cancellation. Merging the subtraction into the library function allows for a different implementation that sidesteps issues related to cancellation. When x is far away from zero, there is no difference from a numerical point of view, and the benefits are negligible (1 FLOP saved).