# Elementary Functions & Stochastic Arithmetic: Lecture Summary and Connections to Exercises

AFAE - Master 2 CCA
Floating-point Arithmetic and Error Analysis

Academic Year 2025/2026

## Contents

# 1 Lecture Overview: Elementary Functions

## 1.1 Main Objectives

The lecture focuses on **how to compute elementary functions** (like exp, sin, log, etc.) on computers using floating-point arithmetic. The key goals are:

1. Understand why we need special algorithms (can't compute exactly)
2. Learn polynomial approximation techniques
3. Master argument reduction strategies
4. Achieve good performance (e.g., exp in 40 cycles)
5. Understand correctly rounded functions
6. Explore automation of libm generation

## 1.2 Why This Matters

- Basic operations $(+, -, \times, /)$ give exact results (rational numbers)
- Elementary functions $(\exp, \sin, \log)$ give **transcendental** results
- Cannot be computed exactly $\Rightarrow$ need approximations
- Used everywhere: bacteria growth, waves, finance, statistics

# 2 Key Concepts and Theorems

## 2.1 Floating-Point Representation

**Definition 1** (Floating-Point Number)**.** A floating-point number in radix $\beta = 2$ with precision $k$ is represented as:

$$x = \pm 2^E \cdot m$$

where:

- $E$ is the exponent (gives order of magnitude)
- $m$ is the significand (gives the digits)
- The set of such numbers is denoted $\mathbb{F}k$

**Property 2** (Precision of Standard Formats)**.**

$$\begin{aligned} \text{binary32 (float):} \quad & -\log_{10}(2^{-24}) \approx 7.2 \text{ decimal digits} \\ \text{binary64 (double):} \quad & -\log_{10}(2^{-53}) \approx 15.9 \text{ decimal digits} \end{aligned}$$

## 2.2 Polynomial Approximation

### 2.2.1 Weierstrass Approximation Theorem

**Theorem 3** (Weierstrass)**.** For any continuous function $f : [a, b] \to \mathbb{R}$ and any $\varepsilon > 0$, there exists a polynomial $p$ such that:

$$\|f - p\|_\infty < \varepsilon$$

This is the fundamental reason we use polynomials!

### 2.2.2 Taylor Polynomials

**Theorem 4** (Taylor Expansion)**.** For a function $f$ with $n + 1$ continuous derivatives:

$$f(x) = \sum_{i=0}^{n} \frac{f^{(i)}(x_0)}{i!} \cdot (x - x_0)^i + \frac{f^{(n+1)}(\xi)}{(n+1)!} \cdot (x - x_0)^{n+1}$$

where $\xi \in [x_0, x]$ and the last term is the Lagrange remainder.

**Problem with Taylor:** Error blows up at boundaries of domain.

### 2.2.3 Interpolation Polynomials

**Theorem 5** (Interpolation Error)**.** For a polynomial $p$ of degree $n$ interpolating $f$ at points $x_j$:

$$f(x) = p(x) + \frac{1}{(n+1)!} f^{(n+1)}(\xi) \prod_{j=0}^{n} (x - x_j)$$

**Key Insight:** Choice of interpolation points $x_j$ affects error!

### 2.2.4 Chebyshev Points

**Definition 6** (Chebyshev Interpolation Points)**.** On interval $[a, b]$, the Chebyshev points that minimize $\left\| \prod_{j=0}^{n} (x - x_j) \right\|_\infty$ are:

$$x_j = a + \frac{b-a}{2} \cdot \left( \cos\left( \frac{2j-1}{2(n+1)} \pi \right) + 1 \right)$$

These cluster near the boundaries, giving better error distribution.

### 2.2.5 Remez Algorithm (Minimax Polynomial)

**Theorem 7** (Chebyshev-La Vallée-Poussin)**.** A polynomial approximation $p$ is optimal (in the $\|\cdot\|_\infty$ norm) if and only if all extrema of the error $f(x) - p(x)$ have the same absolute value.

The **Remez algorithm** iteratively finds this optimal polynomial by:

1. Interpolating $f(x) + (-1)^j \cdot \varepsilon$ at initial points
2. Exchanging points with locations of error extrema
3. Repeating until extrema are equioscillatory

## 2.3 Argument Reduction

**Definition 8** (Argument Reduction)**.** Given $f : \mathbb{F} \to \mathbb{F}$, an argument reduction consists of:

- A **reduction function** $r : \mathbb{F} \to \mathbb{F}^n$ (simple to compute)
- A **reduced function** $g : \mathbb{F}^n \to \mathbb{F}^m$ (computed by polynomial/table)
- A **reconstruction function** $c : \mathbb{F}^m \to \mathbb{F}$ (simple to compute)

such that $f(x) = c(g(r(x)))$ for all $x \in \mathbb{F}$.

### 2.3.1 Example: Exponential Without Table

**Exponential Argument Reduction**

$$e^x = 2^{\log_2(e)\cdot x} = 2^E \cdot 2^{\log_2(e)\cdot x - E} = 2^E \cdot e^{x - \frac{E}{\log_2(e)}} = 2^E \cdot e^r$$

where:

- $E = \lfloor \log_2(e) \cdot x \rceil$ (nearest integer)
- $r = x - \frac{E}{\log_2(e)}$ (reduced argument)
- $|r| \leq \frac{1}{2\log_2(e)} \approx 0.35$

### 2.3.2 Example: Exponential With Table

**Table-Based Reduction**

$$e^x = 2^E \cdot 2^{i \cdot 2^{-w}} \cdot e^r$$

where:

- $k = \lfloor \log_2(e) \cdot x \cdot 2^w \rceil$
- $E = \lfloor k \cdot 2^{-w} \rfloor$ (integer part)
- $i = k \cdot 2^{-w} - E$ (table index)
- $r = x - \frac{k \cdot 2^{-w}}{\log_2(e)}$ (reduced argument)
- $|r| \leq 2^{-w} \cdot \frac{1}{2\log_2(e)}$
- $w$ = number of bits for table indexing

Read $2^i$ from precomputed table with $2^w$ entries.

## 2.4 Correctly Rounded Functions

**Definition 9** (Correct Rounding)**.** Let $f : \mathbb{R}^n \to \mathbb{R}$ and $\circ_k : \mathbb{R} \to \mathbb{F}k$ be a rounding. A function $F : \mathbb{F}k^n \to \mathbb{F}k$ is a **correctly rounded** implementation of $f$ if:

$$\forall x \in \mathbb{F}k^n, \quad F(x) = \circ_k(f(x))$$

**The Table Maker's Dilemma:** We can only compute $\hat{y} = f(x) \cdot (1+\varepsilon)$. How much precision $\varepsilon$ is needed for the worst case?

**Lemma 10** (Worst Case for Exponential - Double Precision)**.** For $f = \exp$ and rounding $\circ_{53} : \mathbb{R} \to \mathbb{F}53$ in double precision, let $D = \mathbb{F}53 \cap [-744.5, 709]$.

Then for all $x \in D \setminus \{0\}$ and all $|\varepsilon| \leq 2^{-159}$:

$$\circ_{53}(f(x) \cdot (1+\varepsilon)) = \circ_{53}(f(x))$$

This means we need **159 bits of precision** for correct rounding of exp in double precision!

## 2.5 Performance Techniques

### 2.5.1 Computing Nearest Integer Without Function Call

**Magic Number Technique**

To compute $\lfloor x \rceil$ (nearest integer) efficiently:

```
double shifter = 6755399441055744.0;  // 2^52 + 2^51
double tmp = x + shifter;                 // rounds to nearest
double nearest = tmp - shifter;
```

Works because for $x$ sufficiently small, $2^{52} + 2^{51} + x$ has ulp $= 1$.

### 2.5.2 Constructing $2^E$ Directly

**Bit Manipulation**

```
int E;
unsigned long long int tmp;
double twoE;

tmp = E + 1023;           // add double precision bias
tmp <<= 52;               // shift to exponent position
twoE = *((double *) &tmp);// interpret as double
```

This directly constructs the floating-point representation!

# 3 Connection to Exercise 4: Stochastic Arithmetic

## 3.1 The Link

The lecture on elementary functions and Exercise 4 are **closely related** through the theme of **numerical accuracy and error analysis**.

### 3.1.1 Shared Concepts

1. **Floating-Point Precision**

- Lecture: Uses binary32 (7.2 digits) and binary64 (15.9 digits) for function implementation
- Exercise 4: Analyzes how cancellation affects these precisions differently

2. **Error Accumulation**
   - Lecture: Five sources of error in function evaluation:
     (a) Error in argument reduction
     (b) Error in table entries
     (c) Approximation error $\|p/f - 1\|_\infty$
     (d) Error in polynomial evaluation
     (e) Error in reconstruction
   - Exercise 4: Catastrophic cancellation as a major source of error

3. **Required Precision for Accuracy**
   - Lecture: Need 159 bits for correctly rounded exp in double precision
   - Exercise 4: Need to understand how many bits are lost to cancellation

## 3.2 Key Theorem Connecting Both

**Theorem 11** (Independence of Accuracy Loss - from Exercise 4)**.** The **loss of accuracy** during a numerical computation is **independent** of the precision used for the floating-point representation.

**Application to Lecture Material:**
When implementing elementary functions, if catastrophic cancellation occurs in:

- The argument reduction step
- The polynomial evaluation
- The reconstruction

Then the number of digits lost will be the **same** whether we use binary32 or binary64!

## 3.3 Cancellation in Function Implementation

### 3.3.1 Example from Exercise 4

In Gaussian elimination with $a = b + 1$ and $c = b - 1$:

$$c - \frac{b^2}{a} = (b - 1) - \frac{b^2}{b + 1} = \frac{-1}{b + 1}$$

For $b = 303$:

$$302 - 302.003289... = -0.003289...$$

Digits lost: $\log_{10}(302/0.003289) \approx 5$ decimal digits

### 3.3.2 Similar Issue in Elementary Functions

In the argument reduction for $\exp(x)$:

$$r = x - \frac{k \cdot 2^{-w}}{\log_2(e)}$$

If $x \approx \frac{k \cdot 2^{-w}}{\log_2(e)}$, we get catastrophic cancellation!
**This is why:**

- The value $\frac{1}{\log_2(e)}$ must be stored with **very high precision**
- The subtraction must be computed carefully
- Multi-precision arithmetic may be needed (as mentioned in lecture)

## 3.4 DSA/CADNA for Function Validation

**Application to libm Testing**

**DSA (Discrete Stochastic Arithmetic)** from Exercise 4 could be used to:

1. **Detect instabilities** in function implementations
   - Run exp, sin, log with CADNA
   - Identify inputs where accuracy is lost
   - These are candidates for "hard to round" cases!
2. **Validate polynomial approximations**
   - Check if polynomial evaluation is stable
   - Detect when coefficients need more precision
3. **Verify argument reduction**
   - Check if reduced argument $r$ has enough accurate digits
   - Warning if cancellation occurs

## 3.5 The Toy Exponential from Lecture

From the lecture code (slide 17 & 48):

```
// About 45 bits of accuracy
double Exp(double x) {
    // Argument reduction
    z = x * TWO_4_RCP_LN_2;
    // ... compute E, idx, r ...
    r = x - t;  // <-- POTENTIAL CANCELLATION HERE!

    // Polynomial approximation
    P = c0 + r*(c1 + r*(c2 + r*(c3 + r*(c4 + r*c5))));
```

```
    // Reconstruction
    y = twoE.d * (tbl * P);
    return y;
}
```

**Question from Exercise 4 perspective:** What happens if:

- $x \approx t$ (near a table boundary)?
- We only have 53 bits precision?
- DSA might warn: "LOSS OF ACCURACY DUE TO CANCELLATION"!

# 4 Key Formulas Summary

## 4.1 Polynomial Approximation

Essential Formulas

**Taylor Polynomial:**

$$p(x) = \sum_{i=0}^{n} \frac{f^{(i)}(x_0)}{i!}(x - x_0)^i$$

**Chebyshev Points (on $[a, b]$):**

$$x_j = a + \frac{b-a}{2}\left(\cos\left(\frac{2j-1}{2(n+1)}\pi\right) + 1\right)$$

**Interpolation Error:**

$$|f(x) - p(x)| \leq \frac{1}{(n+1)!}|f^{(n+1)}(\xi)| \prod_{j=0}^{n} |x - x_j|$$

## 4.2 Argument Reduction for Common Functions

Standard Reductions

**Exponential:**

$$e^x = 2^E \cdot 2^i \cdot e^r$$

where $E$ is integer exponent, $i$ is table index, $|r|$ small.
**Sine (periodicity):**

$$\sin(x) = \sin(x - 2\pi k), \quad k = \left\lfloor \frac{x}{2\pi} \right\rceil$$

**Logarithm:**

$$\log(x) = \log(2^E \cdot m) = E \log(2) + \log(m)$$

where $m \in [1, 2)$ is the significand.

### 4.3 Error Analysis (from Exercise 4)

Cancellation Formula

**Digits Lost in Cancellation:**
For $x \approx y$, computing $x - y$:

$$\text{Digits lost} \approx \log_{10}\left(\frac{|x|}{|x-y|}\right)$$

**This is INDEPENDENT of precision format!**
**Remaining Precision:**

$$\begin{aligned} \text{binary32:} \quad & 7.2 - (\text{digits lost}) \\ \text{binary64:} \quad & 15.9 - (\text{digits lost}) \end{aligned}$$

### 4.4 DSA Accuracy Estimation

CESTAC Method

**Number of Significant Digits:**

$$C_R \approx \log_{10}\left(\frac{\sqrt{N}\,|\bar{R}|}{\sigma \tau_\beta}\right)$$

where:

- $\bar{R} = \frac{1}{N}\sum_{i=1}^{N} R_i$ (mean of $N$ samples with random rounding)
- $\sigma^2 = \frac{1}{N-1}\sum_{i=1}^{N}(R_i - \bar{R})^2$ (variance)
- $\tau_\beta$ = Student's t-distribution value
- $N$ = number of samples (typically 3)

## 5 Practical Implications

### 5.1 For Function Implementation

1. **Choose reduction carefully** to avoid cancellation
   - Store constants like $\frac{1}{\log_2(e)}$ with extra precision
   - Use compensated algorithms if needed
2. **Use fpminimax** for polynomial coefficients
   - Not just real coefficients
   - Need floating-point coefficients that preserve error bounds
3. **Analyze all error sources**

- Approximation error (controllable via degree)
- Rounding error (depends on precision)
- Cancellation (can be catastrophic!)

### 5.2 For Numerical Analysis

1. **Don't trust all displayed digits**
   - Classical FP shows 53 bits even if only 1 bit is reliable
   - DSA/CADNA shows only reliable digits
2. **Higher precision helps but doesn't solve everything**
   - binary64 vs binary32: more buffer, same loss
   - Algorithmic improvements ¿ precision increases
3. **Test systematically**
   - Sampling isn't enough (worst cases are rare!)
   - Need formal proof or exhaustive testing
   - DSA can help identify problems

## 6 Conclusion: Unified View

The Big Picture

**Elementary Functions** teaches us:

- How to implement transcendental functions efficiently
- Use polynomial approximation + argument reduction + tables
- Achieve 40-cycle exp with 53-bit accuracy

**Stochastic Arithmetic (Exercise 4)** teaches us:

- How to detect when accuracy is lost
- Cancellation loses digits independent of precision
- DSA reveals actual accuracy vs. displayed precision

**Together they show:**

- Function implementation is an **error analysis problem**
- Must carefully analyze all operations for stability
- Tools like DSA/CADNA can validate implementations
- Correctly rounded functions require understanding worst cases

# 7 Further Study

## 7.1 Recommended Reading

1. **Muller**, *Elementary Functions, Algorithms and Implementation*, Birkhäuser, 2016
   - Comprehensive reference for function implementation
2. **Abramowitz and Stegun**, *Handbook of Mathematical Functions*
   - Source of remarkable identities for argument reduction
3. **Tang**, “Table-driven implementation of the exponential function”, TOMS 15(2), 1989
   - Classic paper on efficient implementation

## 7.2 Key Takeaways for Exams/Projects

1. Know the **Remez algorithm** and why it’s optimal
2. Understand **argument reduction** examples (especially exp)
3. Remember **cancellation independence theorem**
4. Be able to compute **digits lost** in cancellation
5. Understand the **Table Maker’s Dilemma**
6. Know how to use **bit manipulation** for performance
7. Understand **DSA/CADNA** for validation