Numerical Validation Using the CADNA Library

Practical Work – Exercises 1-7

Alberto Taddei *Giulia Lionetti*

AFAE – Floating-Point Arithmetic and Error Analysis
Master 2 CCA, Sorbonne Université

October 2025

Abstract

This report presents the results of seven numerical exercises designed to demonstrate the importance of numerical validation in scientific computing. Using the CADNA (Control of Accuracy and Debugging for Numerical Applications) library, we analyze catastrophic cancellation, ill-conditioned systems, unstable recurrences, iterative methods, and chaotic systems. The exercises reveal how standard floating-point arithmetic can produce completely incorrect results that appear valid, and demonstrate how CADNA's stochastic arithmetic exposes these hidden errors.

Contents

1 Exercise 1: Rump's Polynomial Evaluation

1.1 Background

We evaluate the polynomial function:

$$f(x, y) = 333.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5.5y^8 + \frac{x}{2y}$$

with $x = 77617$ and $y = 33096$.

The exact result (first 15 digits) is: -0.827396059946821

1.2 Question 1: Float Precision Results

Execution of `rump.c`:

```
res = 7.08931176699221e+29
```

Analysis: The computed result is catastrophically wrong:

- Wrong sign (positive instead of negative)
- Wrong magnitude (10^{29} instead of $\sim 10^{-1}$)
- Error of approximately 10^{29}
- **Zero correct significant digits**

Standard floating-point arithmetic provides no warning that this result is meaningless.

1.3 Question 2: Double Precision Results

Execution of `rumpd.c`:

```
res = 1.17260394005318e+00
```

Analysis: Even with double precision, the result remains completely incorrect:

- Still wrong sign (positive instead of negative)
- Magnitude closer but still wrong (1.17 vs -0.827)
- Error of approximately 2.0
- **Zero correct significant digits**

Increasing precision from float to double did not solve the problem. This demonstrates that numerical catastrophe can occur even in double precision.

1.4 Question 3: CADNA Implementation

Execution of `rump_cad.cc` and `rumpd_cad.cc`:

```
res = @.0
Instabilities: 1 LOSS OF ACCURACY DUE TO CANCELLATION
```

Analysis: CADNA correctly identifies that the result has **no significant digits** (`@.0` = computational zero). Both float and double versions show the same behavior with CADNA.

Instabilities Detected:

- 1 catastrophic cancellation detected
- This occurs when subtracting nearly equal large numbers, destroying all significant digits

1.5 Conclusions

1. **Standard arithmetic fails silently:** Without CADNA, we would accept completely wrong results (7×10^{29} or 1.17) as valid.

2. **Precision is not accuracy:** Increasing from float to double did not help: both give wrong results because the problem is algorithmic, not precision-based.

3. **CADNA reveals the truth:** The `@.0` result correctly indicates that no significant digits remain after the catastrophic cancellation.

4. **Root cause:** The polynomial involves subtracting very large intermediate values that nearly cancel, destroying all accuracy. This is a classic example of catastrophic cancellation.

5. **Practical implication:** This demonstrates why numerical validation is essential: code that compiles and runs without errors can produce completely meaningless results.

2 Exercise 2: Hilbert Matrix Determinant

2.1 Background

We compute the determinant of the 11×11 Hilbert matrix using Gaussian elimination. The determinant is the product of the pivots.

Exact determinant (first 14 digits): $3.0190953344493 \times 10^{-65}$

2.2 Question 1: Standard Computation Results

Execution of `hilbert.c`:

Pivot	Value
Pivot 0	$+1.000000000000000 \times 10^{0}$
Pivot 1	$+8.333333333333331 \times 10^{-2}$
Pivot 2	$+5.555555555555522 \times 10^{-3}$
Pivot 3	$+3.571428571428736 \times 10^{-4}$
Pivot 4	$+2.267573696146732 \times 10^{-5}$
Pivot 5	$+1.431549050481817 \times 10^{-6}$
Pivot 6	$+9.009749236431395 \times 10^{-8}$
Pivot 7	$+5.659970607161749 \times 10^{-9}$
Pivot 8	$+3.551362553328898 \times 10^{-10}$
Pivot 9	$+2.226656943069665 \times 10^{-11}$
Pivot 10	$+1.398301799864147 \times 10^{-12}$
Determinant	$+3.026439382718219 \times 10^{-65}$

Comparison with exact value:

- Computed: $3.0264393827\ldots \times 10^{-65}$
- Exact: $3.0190953344\ldots \times 10^{-65}$
- Only first 2.61 digits are correct

Rigorous accuracy calculation using the theoretical formula:

Error relative:

$$\left|\frac{R-r}{r}\right| = \frac{|3.0264393827 - 3.0190953344|}{3.0190953344} \approx 0.002433$$

Number of correct significant bits:

$$C_R = -\log_2(0.002433) \approx 8.68 \text{ bits}$$

Number of correct decimal digits:

$$\text{Decimal digits} = \frac{C_R}{\log_2(10)} = \frac{8.68}{3.322} \approx 2.61 \text{ digits}$$

Therefore, only approximately **2.61 decimal digits** are reliable, confirming that the standard computation severely underestimates the actual error despite appearing to have 15 digits of precision.

Analysis: The result appears "reasonable" – correct order of magnitude and sign. However, only 2.61 significant digits are accurate. Without CADNA, this loss of accuracy is invisible.

2.3 Question 2: CADNA Implementation

Execution of `hilbert_cad.cc`:

Pivot	Value	Significant Digits
Pivot 0	$0.100000000000000 \times 10^{+1}$	15
Pivot 1	$0.833333333333333 \times 10^{-1}$	15
Pivot 2	$0.55555555555555 \times 10^{-2}$	14
Pivot 3	$0.3571428571428 \times 10^{-3}$	13
Pivot 4	$0.22675736961 \times 10^{-4}$	11
Pivot 5	$0.143154905 \times 10^{-5}$	9
Pivot 6	$0.90097492 \times 10^{-7}$	8
Pivot 7	0.5659970×10^{-8}	7
Pivot 8	0.35513×10^{-9}	5
Pivot 9	0.2226×10^{-10}	4
Pivot 10	0.13×10^{-11}	2
Determinant	0.30×10^{-64}	**2**

Instabilities: No instability detected

Detailed observation: The step-by-step elimination showed:

- Progressive accumulation of rounding errors
- Each elimination step loses approximately 1-2 significant digits
- After 10 steps, only 2 digits remain

2.4 Conclusions

1. **Progressive accuracy loss:** Even without detected instabilities, Gaussian elimination on ill-conditioned matrices causes progressive loss of significant digits.
2. **True accuracy revealed:** CADNA shows the determinant has only 2 reliable digits, confirming that the standard result (appearing to have 15 digits) is misleading.
3. **Ill-conditioning:** The Hilbert matrix is notoriously ill-conditioned. The condition number grows exponentially with matrix size, making numerical computation extremely sensitive to rounding errors.
4. **No instability $\neq$ accurate result:** The absence of detected instabilities does not guarantee accuracy. Gradual accumulation of errors can be equally destructive.
5. **Validation importance:** Standard computation provides a result that looks correct but is accurate to only 2 digits. CADNA makes this limitation explicit.
6. **Limitation of method:** For such ill-conditioned problems, standard Gaussian elimination is inadequate. Higher precision or alternative algorithms would be needed.

3 Exercise 3: Muller's Recurrence Sequence

3.1 Background

We compute the first 30 iterations of the recurrence relation:

$$U_{n+1} = 111 - \frac{1130}{U_n} + \frac{3000}{U_{n-1} \times U_n}$$

with initial conditions $U_0 = 5.5$ and $U_1 = \frac{61}{11}$.

The expected limit is: $U_\infty = 6$

3.2 Question 1: Standard Computation Results

Execution of `muller.c`:

Iteration	Value	Observation
$U(2)$	$+5.590163934426229 \times 10^0$	
$U(3)$	$+5.633431085043981 \times 10^0$	
$U(4)$	$+5.674648620510027 \times 10^0$	
$\vdots$	$\vdots$	Approaching 6
$U(12)$	$+5.898177025615013 \times 10^0$	
$U(13)$	$+5.897965247556456 \times 10^0$	Divergence begins
$U(14)$	$+5.647011084038567 \times 10^0$	
$U(15)$	$+9.683399445297454 \times 10^{-1}$	
$U(16)$	$-5.073216051624674 \times 10^2$	Catastrophic
$U(17)$	$+1.071206352328062 \times 10^2$	
$U(18)$	$+1.003959421894409 \times 10^2$	
$\vdots$	$\vdots$	False convergence
$U(30)$	$+1.000000000000000 \times 10^2$	to 100

Analysis: The sequence does **NOT** converge to 6. Instead:

- Iterations 2-12: Appears to converge toward 6 (values $5.59 \to 5.89$)
- Iteration 13: Sudden change (still ~ 5.89)
- Iterations 14-15: Rapid divergence ($5.64 \to 0.97$)
- Iteration 16: Catastrophic jump to -507
- Iterations 17-30: False "convergence" to 100

"Are the results correct?" No: The sequence converges to 100 instead of 6, which is completely wrong.

3.3 Question 2: CADNA Implementation and Instability Analysis

Execution of `muller_cad.cc`:

Iteration	Value
$U(2)$	$0.55901639344262 \times 10^1$
$U(3)$	$0.5633431085044 \times 10^1$
$U(4)$	$0.56746486205 \times 10^1$
$U(5)$	0.5713329052×10^1
$U(6)$	0.574912092×10^1
$U(7)$	0.57818109×10^1
$U(8)$	0.581131×10^1
$U(9)$	0.58376×10^1
$U(10)$	0.5860×10^1
$U(11)$	0.588×10^1
$U(12)$	0.59×10^1
$U(13)$	`@.0` Loss of all significant digits
$U(14)$	`@.0`
$U(15)$	`@.0`
$U(16)$	0.9×10^2 Spurious "recovery"
$U(17)$	0.99×10^2
$U(18)$	0.999×10^2
$\vdots$	$\vdots$
$U(30)$	$0.100000000000000 \times 10^3$

CRITICAL WARNING: The self-validation detects major problem(s). The results are NOT guaranteed.

Instabilities: 8 numerical instabilities:

- 6 UNSTABLE DIVISION(S)
- 2 UNSTABLE MULTIPLICATION(S)

Identification of instabilities: The instabilities occur at **iterations 12-15** when:

- $U(12)$: Only 2 significant digits remain
- $U(13)$-$U(15)$: Values become `@.0` (computational zero, no significant digits)
- The divisions and multiplications involve non-significant operands

"Are the last iterates reliable?" No: The last iterates (convergence to 100) are not reliable because:

1. The sequence passed through `@.0` (computational zero) before "recovering"
2. Once all significant digits are lost, subsequent values are meaningless
3. The "convergence" to 100 is spurious: it's numerical noise, not a true limit

Why are they unreliable? The sequence became numerically unstable around iteration 12. Small rounding errors, amplified exponentially by the recurrence relation, destroyed all accuracy. The subsequent "recovery" to 100 is an artifact of floating-point arithmetic operating on meaningless values.

3.4 Conclusions

1. **Numerical instability in recurrence relations:** The sequence is numerically unstable. Small rounding errors grow exponentially, eventually destroying all accuracy.
2. **Standard arithmetic misleads:** Without CADNA, the convergence to 100 appears genuine; the values show perfect convergence with 15 digits of precision!

3. **CADNA reveals catastrophe:** The `@.0` values at iterations 13-15 correctly indicate that accuracy was lost. Everything after this point is unreliable.

4. **Instability mechanism:**

   - Divisions by very small differences (UNSTABLE DIVISIONS)
   - Multiplications of imprecise values (UNSTABLE MULTIPLICATIONS)
   - These occur when intermediate values lose significance

5. **False precision:** The final value $U(30) = 100.000000000000$ appears to have 15 correct digits, but CADNA reveals it has **zero** correct digits relative to the true limit of 6.

6. **Practical implication:** For numerically unstable recurrences, results become meaningless after a finite number of iterations, regardless of the arithmetic precision used. Alternative formulations or algorithms are required.

4 Exercise 4: Newton's Method for Polynomial Root

4.1 Background

We compute a root of the polynomial:

$$f(x) = 1.47x^3 + 1.19x^2 - 1.83x + 0.45$$

using Newton's method with initial value $x_0 = 0.5$ and stopping criterion $|x_n - x_{n-1}| < 10^{-12}$.

4.2 Question 1: Standard Computation Results

Execution of `newton.c`:

Selected iterations showing the behavior near convergence:

Iteration	x value	Difference	
$x(1)$	$4.64864864864865 \times 10^{-1}$	3.51×10^{-2}	
$x(2)$	$4.46871334005382 \times 10^{-1}$	1.80×10^{-2}	
$\vdots$	$\vdots$	$\vdots$	
$x(22)$	$4.28571446417224 \times 10^{-1}$	1.76×10^{-8}	
$x(23)$	$4.28571437832772 \times 10^{-1}$	8.58×10^{-9}	
$x(24)$	$4.28571432967630 \times 10^{-1}$	4.87×10^{-9}	
$x(25)$	$4.28571430917781 \times 10^{-1}$	2.05×10^{-9}	
$x(26)$	$4.28571434758445 \times 10^{-1}$	3.84×10^{-9}	Oscillations
$x(27)$	$4.28571433301918 \times 10^{-1}$	1.46×10^{-9}	
$x(28)$	$4.28571431396925 \times 10^{-1}$	1.90×10^{-9}	
$x(29)$	$4.28571428207557 \times 10^{-1}$	3.19×10^{-9}	
$x(30)$	$4.28571403441845 \times 10^{-1}$	2.48×10^{-8}	
$x(31)$	$4.28571415992959 \times 10^{-1}$	1.26×10^{-8}	
$\vdots$	$\vdots$	$\vdots$	
$x(36)$	$4.28571425207827 \times 10^{-1}$	0.00×10^{0}	

Sequence limit: $x \approx 0.428571425\ldots$ (approximately $\frac{3}{7} = 0.42857142857\ldots$)

Number of iterations: 36

Critical observation: In iterations 26-35, the difference oscillates (from 3.84×10^{-9} to 2.48×10^{-8}, then back down), indicating numerical instability near convergence. The final convergence (diff = 0) is artificial.

4.3 Question 2: CADNA Implementation and Analysis

4.3.1 Version 1: Original stopping criterion (diff < eps)

Execution of `newton_cad.cc`:

Key iterations:

Iteration	x value	Difference
$x(1)$	$0.464864864864864 \times 10^{+000}$	$0.35135135135135 \times 10^{-001}$
$x(2)$	$0.44687133400538 \times 10^{+000}$	$0.1799353085948 \times 10^{-001}$
$\vdots$	$\vdots$	$\vdots$
$x(20)$	$0.428571499 \times 10^{+000}$	0.70×10^{-007}
$x(21)$	$0.428571463 \times 10^{+000}$	0.35×10^{-007}
$x(22)$	$0.42857144 \times 10^{+000}$	0.1×10^{-007}
$x(23)$	$0.42857143 \times 10^{+000}$	`@.0`
$x(24)$	$0.42857143 \times 10^{+000}$	`@.0`
$x(25)$	$0.42857143 \times 10^{+000}$	`@.0`
$\vdots$	$\vdots$	`@.0`
$x(100)$	$0.4285714 \times 10^{+000}$	`@.0`

CRITICAL WARNING: The self-validation detects major problem(s). The results are NOT guaranteed.

Instabilities: 480 numerical instabilities

- 76 UNSTABLE DIVISION(S)
- 77 UNSTABLE BRANCHING(S)
- 54 UNSTABLE INTRINSIC FUNCTION(S)
- 273 LOSS(ES) OF ACCURACY DUE TO CANCELLATION(S)

Number of iterations: 100 (reached maximum limit)

Analysis: From iteration 23 onwards, diff becomes `@.0` (computational zero – no significant digits). The stopping criterion diff $< 10^{-12}$ is never satisfied because `@.0` cannot be compared reliably to 10^{-12}. The algorithm continues for 76 useless iterations, accumulating 480 instabilities.

4.3.2 Version 2: Improved stopping criterion (x == y)

Execution with x==y criterion:

Iteration	x value	Difference
$x(1)$	$0.464864864864864 \times 10^{+000}$	$0.35135135135135 \times 10^{-001}$
$\vdots$	$\vdots$	$\vdots$
$x(22)$	$0.42857144 \times 10^{+000}$	0.1×10^{-007}
$x(23)$	$0.42857143 \times 10^{+000}$	`@.0`
$x(24)$	$0.42857143 \times 10^{+000}$	`@.0`

Instabilities: 51 numerical instabilities

- 1 UNSTABLE DIVISION
- 1 UNSTABLE BRANCHING
- 49 LOSS(ES) OF ACCURACY DUE TO CANCELLATION(S)

Number of iterations: 24

4.3.3 Comparison of Stopping Criteria

Criterion	Iterations	Total Instabilities	Unstable Div	Unstable Branch	Cancellations
diff < eps	100	480	76	77	273
x == y	24	51	1	1	49
Reduction	**76%**	**89%**	**99%**	**99%**	**82%**

4.4 Conclusions

1. **Spurious convergence in standard arithmetic:** The standard version shows oscillations near convergence (iterations 26-35) before artificially reaching diff = 0. This is a sign of numerical instability that goes undetected.

2. **Convergence limitation revealed:** CADNA shows that after ∼23 iterations, the difference between successive iterates becomes computational zero (`@.0`). The algorithm has reached the practical limit of float precision (∼6-7 significant digits in the result).

3. **Stopping criterion matters critically:**

   - With diff < eps: Algorithm wastes 76 iterations (iterations 24-100) computing on meaningless values, generating 480 instabilities
   - With x == y: Algorithm stops correctly at iteration 24 when numerical convergence is achieved, generating only 51 instabilities

4. **Root cause:** Near convergence, Newton's method computes very small corrections ($f(x)/f'(x)$). When these become smaller than machine precision, the difference is no longer representable accurately, causing the test diff < eps to fail even though numerical convergence has been reached.

5. **Practical recommendation:** For iterative methods with CADNA, use equality-based stopping criteria (x == y) rather than tolerance-based criteria (diff < eps) to avoid wasting iterations on numerically insignificant computations.

6. **Final accuracy:** The result has approximately 6-7 significant digits (0.428571X), not the 15 digits suggested by the standard version. CADNA makes this limitation explicit.

5 Exercise 5: Gaussian Elimination with Partial Pivoting

5.1 Background

We solve a 4×4 linear system using Gaussian elimination with partial pivoting:

$$\begin{bmatrix} 21 & 130 & 0 & 2.1 \\ 13 & 80 & 4.74 \times 10^8 & 752 \\ 0 & -0.4 & 3.9816 \times 10^8 & 4.2 \\ 0 & 0 & 1.7 & 9 \times 10^{-9} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 153.1 \\ 849.74 \\ 7.7816 \\ 2.6 \times 10^{-8} \end{bmatrix}$$

The system matrix has elements ranging from 9×10^{-9} to 4.74×10^8 (9 orders of magnitude variation).

Exact solution: $\mathbf{x}_{\text{sol}} = [1, 1, 10^{-8}, 1]^T$

5.2 Question 1: Standard Computation Results

Execution of `gauss.c`:

Component	Computed	Exact	Status
x_0	$+6.261988 \times 10^{1}$	$+1.000000 \times 10^{0}$	Wrong
x_1	-8.953979×10^{0}	$+1.000000 \times 10^{0}$	Wrong (and wrong sign)
x_2	$+0.000000 \times 10^{0}$	$+1.000000 \times 10^{-8}$	Wrong
x_3	$+9.999999 \times 10^{-1}$	$+1.000000 \times 10^{0}$	Correct

Analysis: Three out of four solution components are completely wrong:

- x_0: Off by factor of 62, wrong by 6100%
- x_1: Wrong sign, magnitude error of 900%
- x_2: Lost entirely (computed as 0 instead of 10^{-8})
- x_3: Only correct component

Are the results correct? No: The algorithm appears to work (no runtime errors) but produces catastrophically wrong results for a severely ill-conditioned system.

5.3 Question 2: CADNA Implementation with Detailed Analysis

Execution of `gauss_cad.cc` with debugging prints:

5.3.1 Critical Moment at Step 1

Matrix after step 1:

$a[0][*]$	$a[1][*]$	$a[2][*]$	$a[3][*]$
0.210×10^{2}	$\cdots$	$\cdots$	$\cdots$
0.130×10^{2}	-0.476×10^{0}	-0.995×10^{9}	$\cdots$
0.000×10^{0}	-0.400×10^{0}	`@.0`	$\cdots$
0.000×10^{0}	0.000×10^{0}	0.170×10^{1}	$\cdots$

Key observation: $a[2][2] =$ `@.0` (Computational Zero: Critical Instability)

5.3.2 Pivoting Decision

Step 2: Pivot selection

- Pivot max $= 0.1700000 \times 10^{1}$ at row 3
- **Swapping rows 2 and 3**: CADNA-guided pivot correction
- New pivot element $a[2][2] = 0.1700000 \times 10^{1}$

5.3.3 Final Results

Component	Computed	Exact	Sig. Digits	Status
x_0	0.99×10^{0}	0.100×10^{1}	2	Correct
x_1	0.100×10^{1}	0.100×10^{1}	3	Correct
x_2	0.999999×10^{-8}	0.100×10^{-7}	6	Correct
x_3	0.100000×10^{1}	0.100×10^{1}	6	Correct

Instabilities: 3 numerical instabilities

- 1 UNSTABLE BRANCHING
- 1 UNSTABLE INTRINSIC FUNCTION
- 1 LOSS OF ACCURACY DUE TO CANCELLATION

All four results are now correct.

5.4 Explanation of Differences Between Standard and CADNA

5.4.1 Key Moment at Step 1

Without CADNA:

- Element $a[2][2]$ after elimination $\approx 3.98 \times 10^8$ (appears as a valid large number)
- Standard pivoting considers this a valid pivot
- Uses this corrupted value for elimination
- Errors propagate catastrophically through back-substitution

With CADNA:

- Element $a[2][2]$ is recognized as `@.0` (computational zero: no significant digits)
- The value 3.98×10^8 is revealed to be numerical garbage
- Pivoting correctly identifies this instability
- Swaps rows 2 and 3, choosing $a[3][2] = 1.7$ as pivot instead
- This correction prevents error propagation

5.4.2 Why Does CADNA Give Better Results?

CADNA uses **stochastic arithmetic** (three parallel computations with random rounding modes):

1. **Breaks systematic error correlations:** In standard arithmetic, rounding errors accumulate systematically in one direction. CADNA's randomization breaks these correlations.
2. **Reveals true accuracy:** Values that appear to have 15 digits but actually have 0 significant digits are exposed as `@.0`.
3. **Enables better algorithmic decisions:** The pivoting strategy can make informed choices based on true numerical significance, not just magnitude.

5.4.3 Instability Analysis

- **1 UNSTABLE BRANCHING:** The `if(ll != i)` test during pivoting involves comparing indices based on values near computational zero
- **1 UNSTABLE INTRINSIC:** `fabs()` applied to a non-significant value during pivot selection
- **1 CANCELLATION:** Subtraction during back-substitution when values are nearly equal

5.5 Conclusions

1. **Silent catastrophic failure:** Standard Gaussian elimination fails completely on this ill-conditioned system without any warning. The code runs successfully but produces garbage.

2. **CADNA corrects the results:** With stochastic arithmetic, the same algorithm produces correct results (within 2-6 significant digits). This is not just diagnostic: CADNA actually improves the numerical outcome.

3. **Mechanism of improvement:** By randomizing rounding errors, CADNA:

   - Prevents systematic error accumulation
   - Exposes values that have lost all significance (`@.0`)
   - Enables the pivoting strategy to make better choices

4. **System characteristics:** The matrix elements span 9 orders of magnitude (10^{-9} to 10^{8}), creating extreme ill-conditioning. Even small rounding errors get amplified catastrophically.

5. **Accuracy limitations:** Even with CADNA, the results have only 2-6 significant digits, revealing the fundamental limitation of float precision for this problem. Double precision would be necessary for better accuracy.

6. **Critical insight:** This exercise demonstrates that CADNA can actively improve numerical results by breaking harmful error correlations that plague standard deterministic arithmetic.

6 Exercise 6: Jacobi Iteration

6.1 Background

We solve a 20×20 linear system using Jacobi iteration with stopping criterion:

$$\max_j |x_j^{\text{new}} - x_j^{\text{old}}| < \varepsilon$$

Testing with various tolerance values ε.

6.2 Question 1: Standard Computation with Various ε

Results:

ε	Iterations	Final errors (range)	Converged?
10^{-4}	1000	7.9×10^{-4} to 5.4×10^{-3}	NO (max iter)
10^{-3}	35	1.2×10^{-4} to 7.0×10^{-3}	YES
5×10^{-3}	30	3.7×10^{-4} to 1.6×10^{-2}	YES
10^{-2}	28	1.8×10^{-3} to 3.0×10^{-2}	YES

Are the results correct? Mathematically yes: the solutions are accurate to within $\sim 10^{-3}$ to 10^{-2}.

Number of iterations: Varies with ε: from 28 ($\varepsilon = 10^{-2}$) to 1000+ ($\varepsilon = 10^{-4}$).

Critical observations:

1. With $\varepsilon = 10^{-4}$: Method does NOT converge in 1000 iterations. Final errors ($\sim 10^{-3}$) are $10\times$ larger than the tolerance.

2. With larger ε: Converges, but final errors often exceed the tolerance. For example, with $\varepsilon = 10^{-2}$, some errors reach 3×10^{-2}.

3. Inconsistency: The stopping criterion (based on successive difference) does not guarantee that the true error meets the tolerance.

6.3 Question 2: CADNA Implementation and Analysis

6.3.1 Initial CADNA Results (ε from 10^{-4} to 10^{-2})

Execution of `jacobi_cad.cc`:

ε	Iterations	Final anorm	Instabilities
10^{-2}	30	`@.0`	330
5×10^{-3}	31	0.0×10^{0}	379
10^{-3}	31	0.0×10^{0}	379
10^{-4}	31	0.0×10^{0}	376

Number of iterations: Approximately 30-31, **CONSTANT regardless of ε value**
Anorm value: `@.0` or 0.0×10^{0} (computational zero)
Is the ε value taken into account? No. The tolerance is completely ignored. All four runs stop at ∼31 iterations regardless of whether $\varepsilon = 10^{-2}$ or 10^{-4}.

6.3.2 Why?

The stopping criterion is `if (anorm < eps)`, where anorm $= \max_j |x_j^{\text{new}} - x_j^{\text{old}}|$.
What happens:

1. Iterations 1-25: Normal convergence, anorm decreases from large values to $\sim 10^{-3}$
2. Iterations 26-30: anorm reaches 10^{-4} to 10^{-5} (near float precision limit)
3. Iteration ∼31: CADNA detects that anorm has lost all significant digits → anorm becomes `@.0`
4. Test `anorm < eps` becomes true (because `@.0` is considered "smaller" than any finite value)
5. Loop terminates **not because tolerance is met, but because precision is exhausted**

The paradox: The algorithm stops "successfully" but for the wrong reason: it stopped because it ran out of numerical accuracy, not because it satisfied the tolerance requirement.

6.3.3 Instability Pattern

- ∼69-114 UNSTABLE BRANCHING: Comparisons involving non-significant values
- ∼16-35 UNSTABLE INTRINSIC: `fabs()` operations on computational zeros
- ∼203-230 CANCELLATIONS: Subtractions in Jacobi iteration near convergence

6.3.4 Improved Stopping Criterion

Modified code:

```
if (anorm < eps || anorm == 0.0) {
  if (anorm == 0.0)
    printf("WARNING: Reached numerical precision limit\n");
  else
    printf("Converged to tolerance\n");
  break;
}
```

Results with improved criterion:

ε	Iterations	Final anorm	Message	Instabilities
10^{-2}	30	0.5×10^{-2}	Converged	288
5×10^{-3}	32	`@.0`	Converged (but `@.0`!)	373
10^{-3}	33	9.8×10^{-4}	Converged	418
10^{-4}	37	0.0	**WARNING: Precision limit**	627

Key improvements:

1. With $\varepsilon \geq 10^{-3}$: Genuine convergence achieved before precision exhaustion
2. With $\varepsilon = 10^{-4}$: Explicit warning that tolerance cannot be met
3. Iteration count now varies with ε (not constant)
4. Distinguished between "true convergence" and "precision limit reached"

Instability growth: As tolerance gets stricter, instabilities increase exponentially:

- $\varepsilon = 10^{-2}$: 288 instabilities
- $\varepsilon = 10^{-4}$: 627 instabilities (2.2× increase)

This occurs because more iterations near convergence involve operations on increasingly small (and less significant) numbers.

6.4 Conclusions

1. **Convergence illusion in standard arithmetic:**
   - With $\varepsilon = 10^{-4}$: Appears to "need" 1000+ iterations but never actually converges
   - With $\varepsilon \geq 10^{-3}$: Appears to converge, but final errors often exceed the tolerance
   - The stopping criterion does not guarantee accuracy
2. ε **is not respected with simple CADNA implementation:** All runs stop at ∼31 iterations because that's when float precision is exhausted, not because the tolerance is met. The value of ε (whether 10^{-2} or 10^{-4}) makes no difference.
3. **Fundamental limitation revealed:** In float precision, Jacobi iteration cannot converge better than ∼30-31 iterations for this problem. After that, successive differences become computational zeros, regardless of the theoretical tolerance requirement.
4. **Why** ε **is ignored:** The test `anorm < eps` becomes trivially true when anorm loses significance (`@.0`), making ε irrelevant. This is "false convergence": the algorithm stops for the wrong reason.
5. **Improved criterion distinguishes genuine vs. spurious convergence:**
   - For $\varepsilon \geq 10^{-3}$: True convergence is possible in float precision
   - For $\varepsilon < 10^{-3}$: Precision limit is reached before tolerance: algorithm correctly warns
6. **Practical implications:**
   - Requesting $\varepsilon = 10^{-4}$ in float precision is unrealistic: use double precision instead
   - Traditional tolerance-based stopping criteria are unreliable near machine precision
   - CADNA-aware stopping criteria are essential for iterative methods
7. **Instability accumulation:** Attempting to converge to tighter tolerances dramatically increases instabilities (from 288 to 627), indicating that the computation becomes increasingly unstable near the precision limit.
8. **Key insight:** The "convergence" reported by standard implementations may be an artifact of precision exhaustion rather than true convergence. CADNA exposes this distinction.

7 Exercise 7: Logistic Iteration (Chaotic System)

7.1 Background

We compute the logistic iteration:

$$x_{n+1} = a \cdot x_n(1 - x_n)$$

with $a = 3.6$ and $x_0 = 0.6$.

An equivalent mathematical form is:

$$x_{n+1} = \frac{a}{4} - a\left(x_n - \frac{1}{2}\right)^2$$

The sequence is **chaotic** (not convergent).

7.2 Question 1: Comparison of Two Equivalent Formulas

Execution with Formula 1: $x = a \cdot x \cdot (1 - x)$

Iteration	Value
$i = 50$	$3.875950637036066 \times 10^{-01}$
$i = 100$	$4.564074671150947 \times 10^{-01}$
$i = 150$	$4.062664991348743 \times 10^{-01}$
$i = 200$	$5.761024866343428 \times 10^{-01}$

Execution with Formula 2: $x = a/4 - a \cdot (x - 0.5)^2$

Iteration	Value
$i = 50$	$3.875950636897043 \times 10^{-01}$
$i = 100$	$4.564083376315182 \times 10^{-01}$
$i = 150$	$4.261690127424104 \times 10^{-01}$
$i = 200$	$3.762927173332673 \times 10^{-01}$

Comparison:

Iteration	Formula 1	Formula 2	Difference	Relative Error
50	0.38759506370	0.38759506368	$\sim 10^{-10}$	$\sim 10^{-10}$
100	0.45640746711	0.45640833763	$\sim 10^{-5}$	$\sim 10^{-5}$
150	0.40626649913	0.42616901274	0.020	4.7%
200	0.57610248663	0.37629271733	**0.200**	**53%**

Analysis: The two **mathematically equivalent** formulas produce **completely different** results after 200 iterations:

- Iteration 50: Nearly identical (10^{-10} difference)
- Iteration 100: Small divergence (10^{-5} difference)
- Iteration 150: Significant divergence (0.02 difference, ∼5% error)
- Iteration 200: **Total divergence** (0.20 difference, 53% error)

Why? The logistic map is **chaotic** – it has sensitive dependence on initial conditions. The two formulas use different arithmetic operations (multiplication vs. subtraction and squaring), producing slightly different rounding errors. These tiny differences ($\sim 10^{-10}$ initially) are exponentially amplified by the chaotic dynamics.

Key observation: After ∼100 iterations, the two formulas have **nothing in common**. Both computations started identically and used mathematically equivalent expressions, yet arrived at completely different values. Neither result can be trusted.

7.3 Question 2: CADNA Implementation and Stopping Criterion

7.3.1 Execution Without Intelligent Stopping Criterion

With CADNA (both formulas, 200 iterations):

Formula 1:

Iteration	Value (Significant Digits)
$i = 50$	0.3875950636×10^0 (10 digits)
$i = 100$	0.45640×10^0 (5 digits)
$i = 150$	0.4×10^0 (1 digit)
$i = 200$	`@.0` (0 digits)

Formula 2:

Iteration	Value (Significant Digits)
$i = 50$	$0.38759506369 \times 10^0$ (11 digits)
$i = 100$	0.456407×10^0 (6 digits)
$i = 150$	0.41×10^0 (2 digits)
$i = 200$	`@.0` (0 digits)

Instabilities: 47 total

- 10 UNSTABLE MULTIPLICATION(S)
- 37 UNSTABLE POWER FUNCTION(S)

CRITICAL WARNING: The self-validation detects major problem(s). Results are not guaranteed.

Description of results:

1. **Progressive accuracy loss:** Both formulas show systematic loss of significant digits:
   - Iteration 50: ∼10-11 digits reliable
   - Iteration 100: ∼5-6 digits reliable
   - Iteration 150: 1-2 digits reliable
   - Iteration 200: 0 digits reliable (`@.0`)
2. **Same pattern for both formulas:** Despite using different operations, both formulas decay to `@.0` at approximately the same rate, confirming that the problem is intrinsic to the chaotic system, not to the formula choice.
3. **Exponential error growth:** Each iteration loses approximately 1-2 bits of accuracy. After ∼150 iterations, all meaningful information is destroyed by accumulated rounding errors.
4. **All iterates after ∼150 are meaningless:** The values computed in iterations 150-200 are pure numerical noise with no connection to the true mathematical sequence.

7.3.2 What Should Be the Stopping Criterion?

Recommended criterion:

```
1 if (x == 0.0 || x == x_old) break;
```

This stops when:

1. **x == 0.0**: Current value has lost all significant digits (became `@.0`)
2. **x == x_old**: Value no longer changes numerically (stagnation)

Both conditions indicate that further iterations are meaningless.

7.3.3 Results with Intelligent Stopping Criterion

Execution with stopping criterion:

Formula 1:

```
i=50  x = 0.3875950636E+000
i=100 x = 0.456407E+000
i=150 x = 0.4E+000
STOPPED at iteration 158: x = @.0
Reason: x lost all significant digits
```

Formula 2:

```
i=50  x = 0.3875950636E+000
i=100 x = 0.45640E+000
i=150 x = 0.4E+000
STOPPED at iteration 156: x = @.0
Reason: x lost all significant digits
```

Instabilities: Only 5 total (all UNSTABLE POWER FUNCTION)
What do you obtain with this new stopping criterion?

1. **Automatic termination at precision limit:**
   - Formula 1 stops at iteration 158 (not 200)
   - Formula 2 stops at iteration 156 (not 200)
   - **42-44 useless iterations avoided** ($\sim$21% of total)
2. **Dramatic instability reduction:**
   - Without criterion: 47 instabilities
   - With criterion: 5 instabilities
   - **89% reduction** in detected instabilities
3. **Same final result:** Both stop with $x =$ `@.0`, correctly indicating that the value is numerically meaningless
4. **Computational efficiency:** Stops exactly when numerical accuracy is exhausted, wasting no further resources
5. **Clear diagnostic:** Explicit message that precision limit was reached, not convergence achieved

7.4 Conclusions

1. **Chaos and floating-point arithmetic are incompatible:**
   - Two mathematically equivalent formulas $\rightarrow$ completely different numerical results
   - After $\sim$100-150 iterations $\rightarrow$ all results are unreliable
   - The "butterfly effect" amplifies rounding errors exponentially
2. **Prediction horizon exposed:** CADNA reveals that for this chaotic system in double precision, there exists a **practical prediction limit of $\sim$150-160 iterations**. Beyond this, computation produces only amplified noise, not meaningful predictions.
3. **False precision without CADNA:** Standard arithmetic displays 15 digits at iteration 200, creating the illusion of accuracy. CADNA reveals these are 0 significant digits (`@.0`).
4. **Formula equivalence is meaningless in chaos:** Mathematical equivalence does not imply numerical equivalence in chaotic systems. Different formulations diverge completely due to tiny rounding differences.

5. **Stopping criterion is essential:**
    - Fixed iteration count (200) wastes ∼21% of iterations on garbage
    - CADNA-based criterion stops exactly when accuracy is exhausted
    - Reduces instabilities by 89%
6. **Implications for chaotic systems** (weather, turbulence, financial markets):
    - Long-term predictions are numerically impossible, not just mathematically difficult
    - There exists a finite "numerical prediction horizon".
    - Beyond this horizon, predictions are meaningless regardless of algorithm quality
7. **CADNA's critical role:** Without CADNA, chaotic computations appear to produce valid results indefinitely. With CADNA, the actual validity limit is exposed, preventing false conclusions based on numerical garbage masquerading as data.
8. **Key message:** This exercise perfectly demonstrates why numerical validation is not optional for scientific computing – it distinguishes between mathematical truth and numerical fiction.

8 General Conclusions

These exercises demonstrate that **numerical validation is not optional**: it is a fundamental requirement for trustworthy scientific computing. CADNA transforms the invisible problem of floating-point error into a visible, manageable aspect of algorithm design. Without such tools, we risk building elaborate computational edifices on foundations of numerical sand.

The distinction CADNA reveals – between what appears to be computed and what is actually computable – is perhaps the most important lesson in numerical analysis:

Not everything that can be calculated should be trusted.