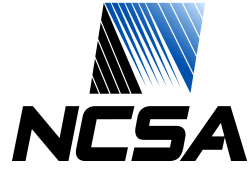
Round-off error and silent soft error propagation in exascale applications

Vincent Baudoui (Argonne/Total SA)
vincent.baudoui@gmail.com
Franck Cappello (Argonne/INRIA/UIUC-NCSA)
Georges Oppenheim (Paris-Sud University)
Philippe Ricoux (Total SA)



Error management in exascale applications

- Stability against round-off errors
- Resilience to hardware errors

Objectives

- Be able to validate simulation results
- Build a model for error propagation

Exascale particularities

- High-dimensional problems
- Different algorithms (e.g. block algorithms)
- Possibly non-deterministic order of operations (e.g. process data as soon as available)

Outline

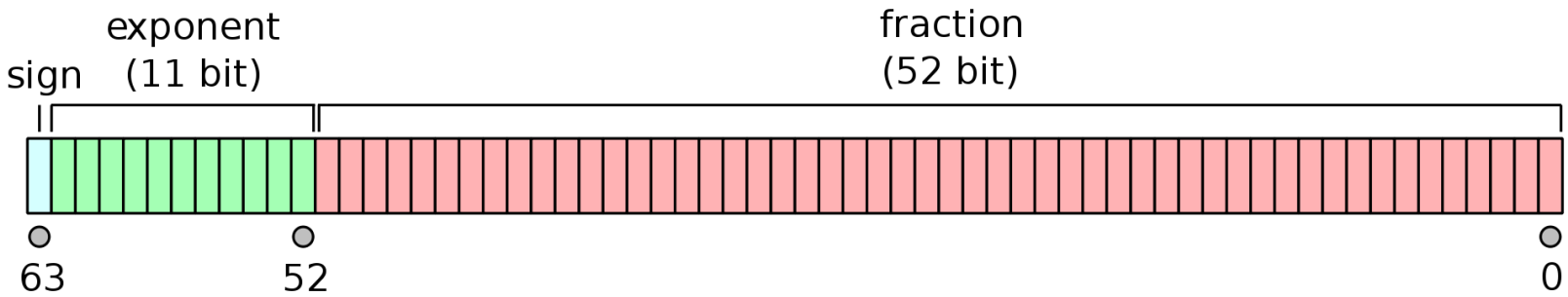
1. Round-off error bounds in high dimensions
2. Code sensitivity analysis strategies
3. Impact of non-determinism

Outline

1. Round-off error bounds in high dimensions
2. Code sensitivity analysis strategies
3. Impact of non-determinism

Real number IEEE 754 floating point encoding

64 bit double precision: $x = \pm 1.f \times 2^{e-1023}$ (for normal numbers)



Limited accuracy (machine precision $u \simeq 10^{-16}$)
$\rightarrow$ round-off errors

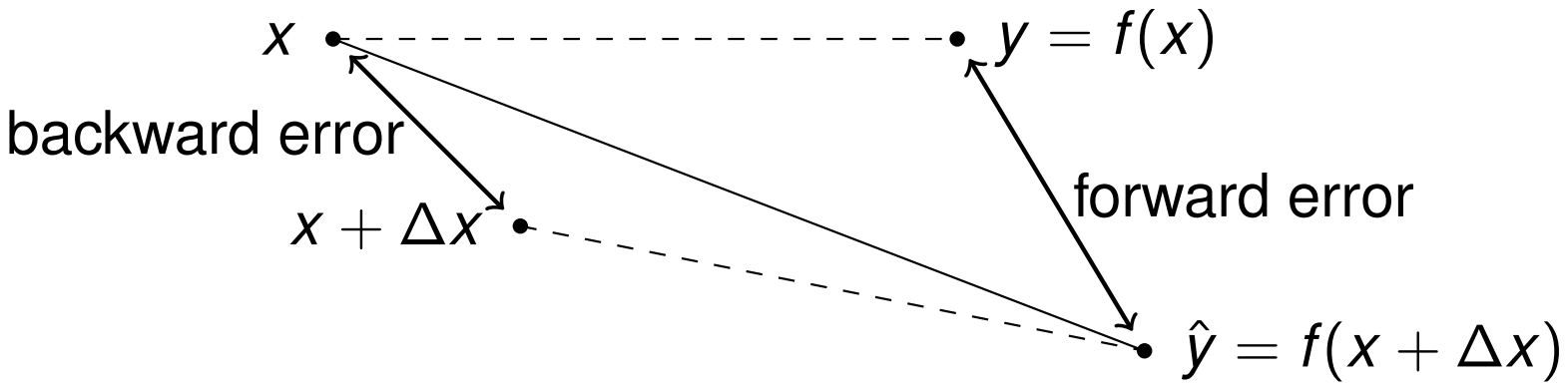
Limited-precision arithmetic

$fl(a \text{ op } b) = (a \text{ op } b)(1+\delta)$ with $|\delta| \leq u$
$\rightarrow$ not necessarily associative nor distributive

Example: $10^{17} - 10^{17} + 1 = 1$, but $10^{17} + 1 - 10^{17} = 0$
$\rightarrow$ importance of data order *(can change with the number of processors in MPI_Reduce for instance)*

If $\hat{y}$ is the computed approximation of $y = f(x)$,
we would like to have $\dfrac{|\hat{y} - y|}{|y|} \approx u$

Forward and backward errors



Forward: error on the result $E = |\hat{y} - y|$

Backward: perturbation in the data $\epsilon = \Delta x$

(for what set of data have we actually solved our problem?)

$$\text{forward error} \lesssim \text{condition number} \times \text{backward error}$$

Higham, N. J. (2002), Accuracy and stability of numerical algorithms

1. Round-off error bounds in high dimensions

Analytical worst-case bounds for linear algebra and some non-linear problems

LU decomposition

Solve $Ax = b$ using LU decomposition

Relative backward error R_n:

$$(A+\Delta A)\hat{x} = b \text{ with } \frac{||\Delta A||_\infty}{||A||_\infty} = \frac{||A\hat{x} - b||_\infty}{||A||_\infty ||\hat{x}||_\infty} = R_n \leq \frac{3n^3 u}{1 - 3nu}\rho_n$$

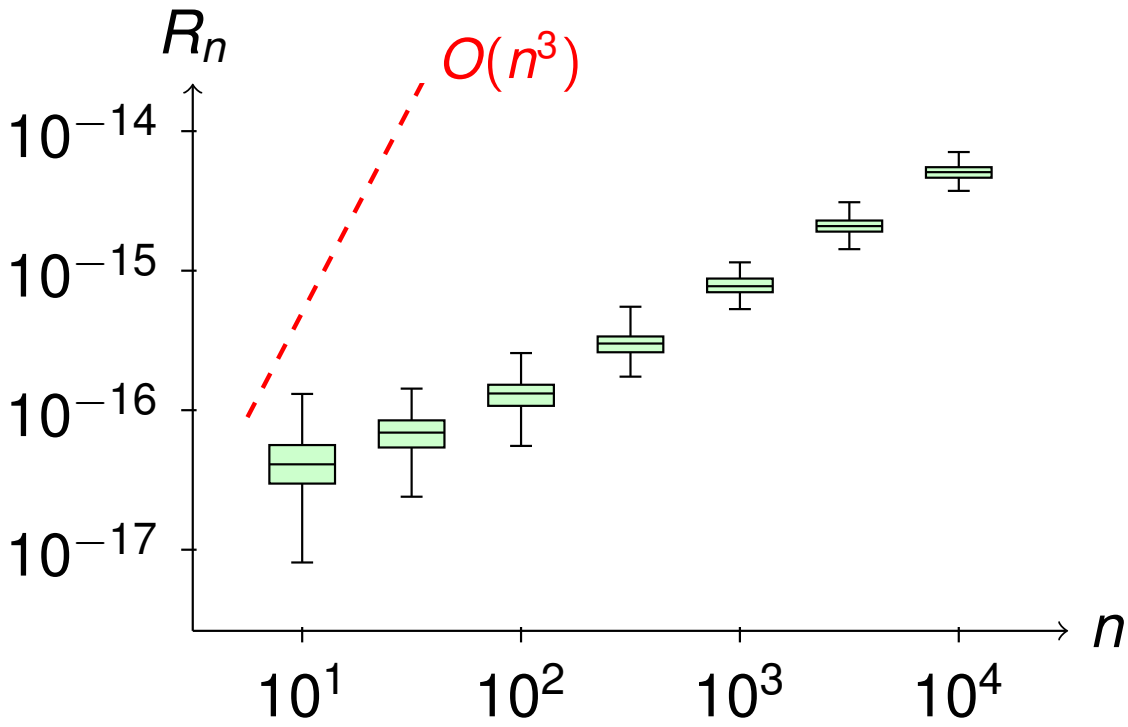
and growth factor $\rho_n = \dfrac{\max\limits_{i,j,k} |a_{ij}^{(k)}|}{\max\limits_{i,j} |a_{ij}|} \leq 2^{n-1}$

1. Round-off error bounds in high dimensions

Experiments on LU decomposition

Scalapack partitioned algorithm with partial pivoting

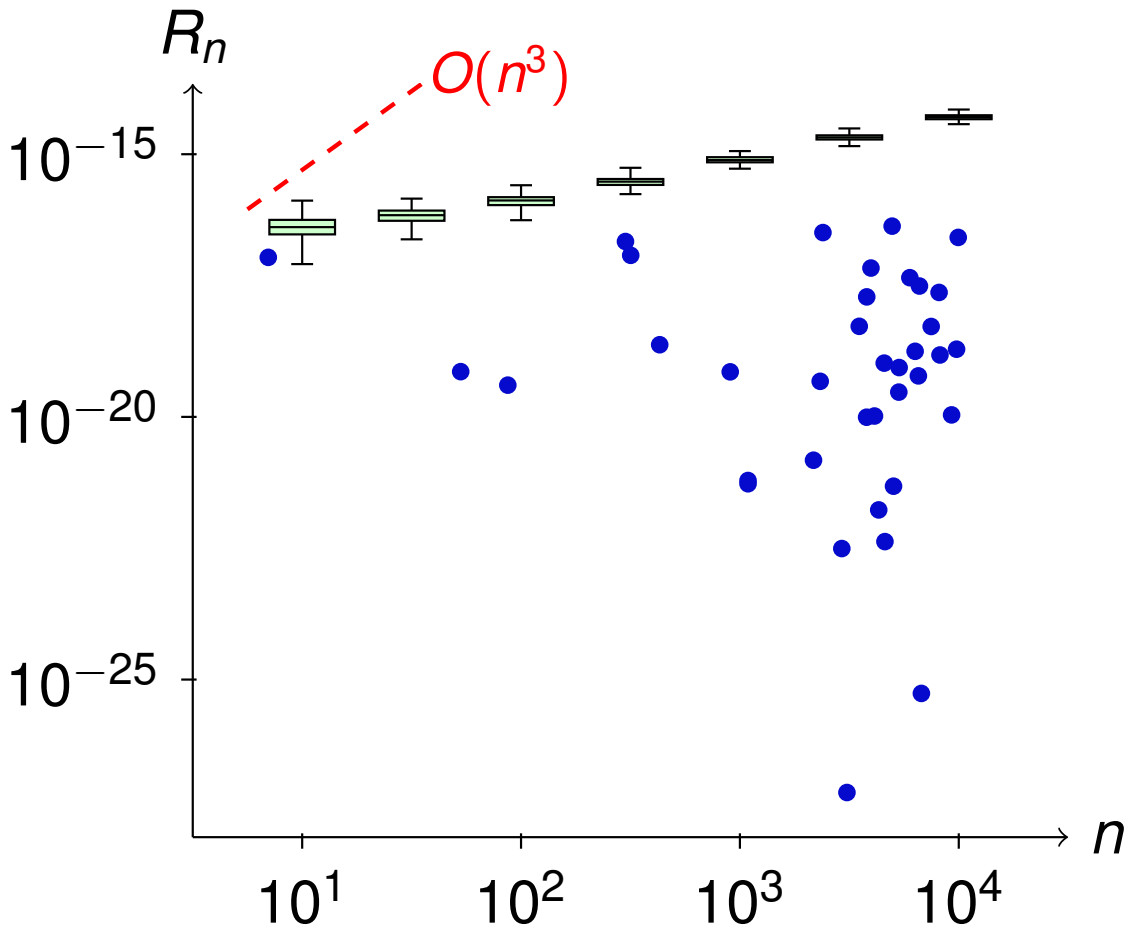
$A, b \sim \mathcal{N}(0, I)$, 250 repeats for each n



$\rightarrow$ pessimistic error bound

Experiments on LU decomposition

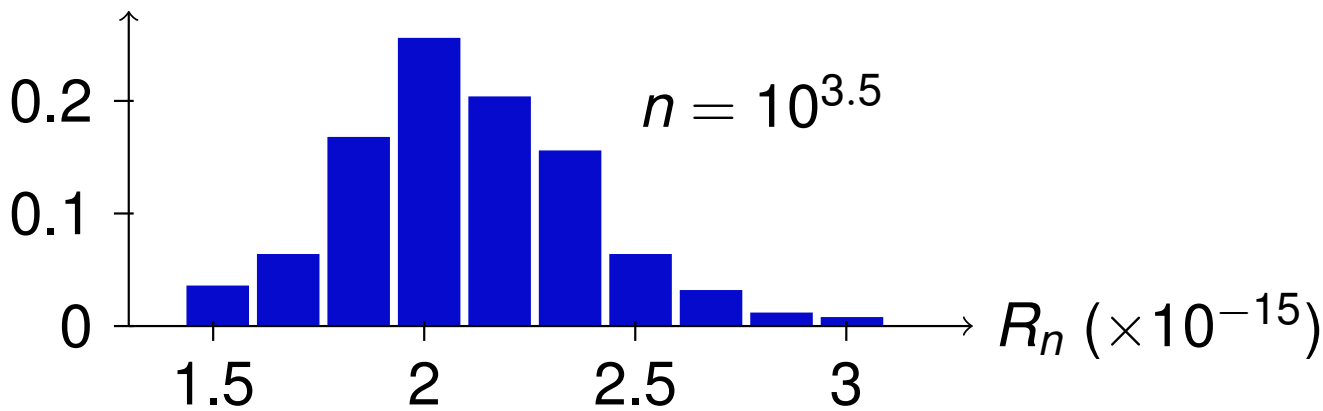
37 problems from the *University of Florida sparse matrix collection*



1. Round-off error bounds in high dimensions

Existing analytical bounds not useful in practice,
especially for high-dimensional problems
$\rightarrow$ rule of thumb: replace n by $\sqrt{n}$ for a more realistic estimate

Express probabilistic bounds!



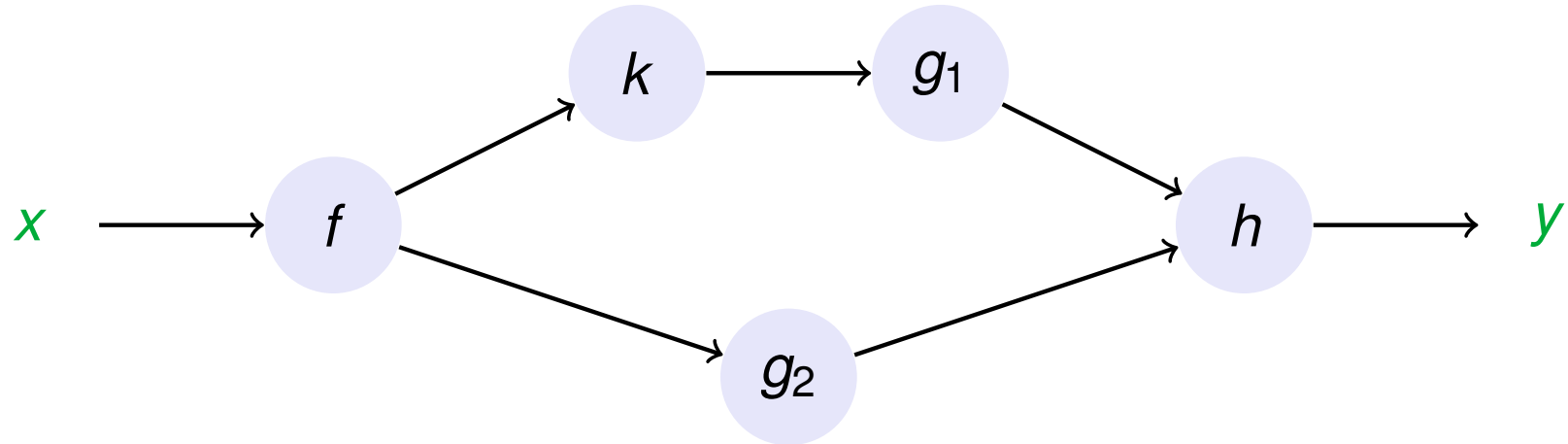
Collaboration with Serge Gratton (CERFACS)

Outline

1. Round-off error bounds in high dimensions
2. Code sensitivity analysis strategies
3. Impact of non-determinism

2. Code sensitivity analysis strategies

Analytical bounds for basic problems only
$\rightarrow$ not enough to study more complex codes



Code sensitivity analysis strategies

- Analyze complex computation accuracy
- Locate sensitive sections of a code

2. Code sensitivity analysis strategies

Experimental accuracy analysis

1. Repeat computation with increasing precision
2. Repeat with different rounding modes:

Rounding	$f(1-u/2)$	$f(1+u)$
to nearest	0.5000000041109161	0.4999999960408469
to zero	0.5000000041036400	0.5000000065775587
up	0.5001221042336121	**0.999999552965182**
down	0.5000000041036401	0.5000000065775587

Kahan, W. (2006), How futile are mindless assessments of roundoff in floating-point computation

2. Code sensitivity analysis strategies

3. Repeat with random rounding:

CADNA: forward error oriented approach

Jézéquel, F. & Chesneaux, J. M. (2008), CADNA: a library for estimating round-off error propagation

Run code 3 times perturbing the result r of every operation:

$$\text{Perturb}(r) = \begin{cases} r & \text{with probability } 1/2 \\ r + \text{one}(r) & \text{with probability } 1/4 \\ r - \text{one}(r) & \text{with probability } 1/4 \end{cases}$$

(where one(r) is of order of the least significant bit of r)

$\rightarrow$ insight on the number of significant digits of the result

4. Repeat with slightly different input data:

PRECISE: data perturbation for backward error analysis

Chaitin-Chatelin, F. & Frayssé, V. (1996), Lectures on finite precision computation

2. Code sensitivity analysis strategies

Accuracy estimation through statistical analysis:

- not foolproof (but rigorous analysis is often infeasible)
- can be computationally expensive (numerous repeats)
- for round-off errors only

→ **looking for new approaches**

→ **consider silent soft errors too**

Silent soft errors

Large scale systems $\rightarrow$ increased hardware error probability

Some errors are corrected, but others can propagate silently

Random data bit flips due to silent soft errors:

$$\text{Perturb}(b_k b_{k-1} \ldots b_1) = \begin{cases} (b_k b_{k-1} \ldots b_1) & \text{with probability } 1-p \\ (\overline{b_k} b_{k-1} \ldots b_1) & \text{with probability } p/k \\ (b_k \overline{b_{k-1}} \ldots b_1) & \text{with probability } p/k \\ \vdots & \\ (b_k b_{k-1} \ldots \overline{b_1}) & \text{with probability } p/k \end{cases}$$

Errors of variable magnitude (can affect the exponent bits)

2. Code sensitivity analysis strategies

Locate sensitive sections of a code:

- Split up code into a graph
- Analyze parallelism: similar computations performed in parallel can be used for a statistical analysis
- Use automatic differentiation to find sections that will magnify errors

Locate sensitive input data: partition input space

Collaboration with Jean Utke and Stefan Wild (Argonne)

Outline

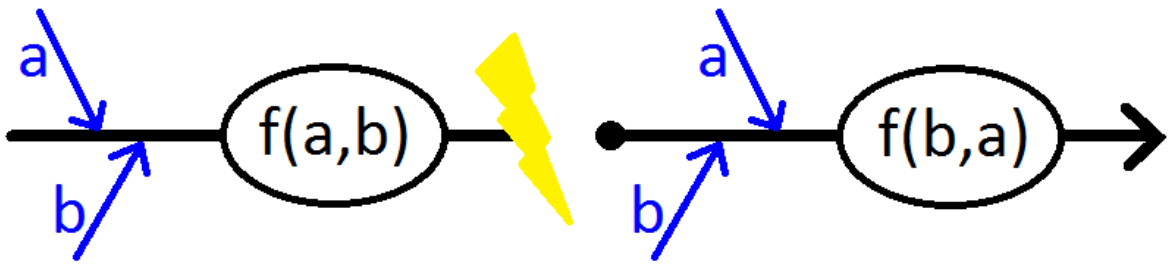
1. Round-off error bounds in high dimensions
2. Code sensitivity analysis strategies
3. Impact of non-determinism

3. Impact of non-determinism

In parallel applications, **data may arrive in random order** (delays in message passing between processes)

If data is processed in order of arrival, results can be non-deterministic because of round-off errors

$\rightarrow$ what happens in case of failure if message order is not stored at checkpoints?



Example: 2D Jacobi

15x15 grid X with initial state X^0

At every step $n+1$, each cell value $x_{i,j}$ is computed as the mean of its 4 neighbors:

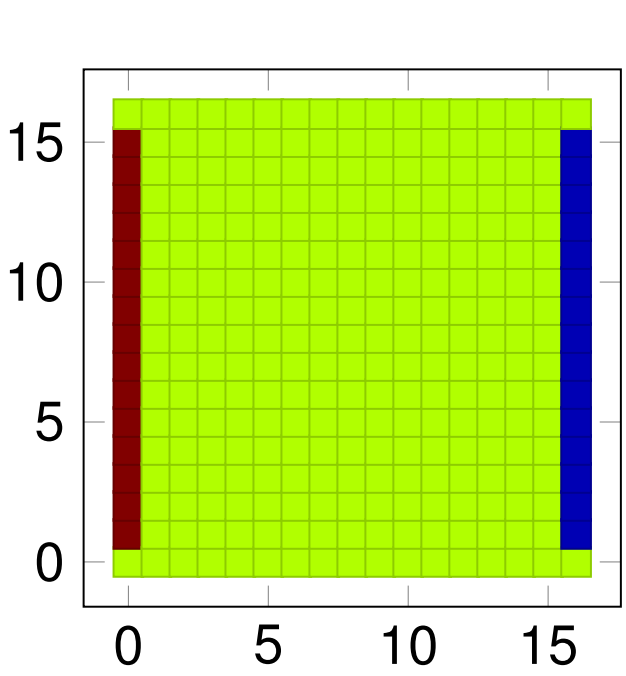
$$\begin{aligned} x_{i,j}^{n+1} &= f(x_{i-1,j}^n, x_{i+1,j}^n, x_{i,j-1}^n, x_{i,j+1}^n) \\ &= \frac{x_{i-1,j}^n + x_{i+1,j}^n + x_{i,j-1}^n + x_{i,j+1}^n}{4} \end{aligned}$$

$\rightarrow$ Markov process: $X^{n+1} = F(X^n)$

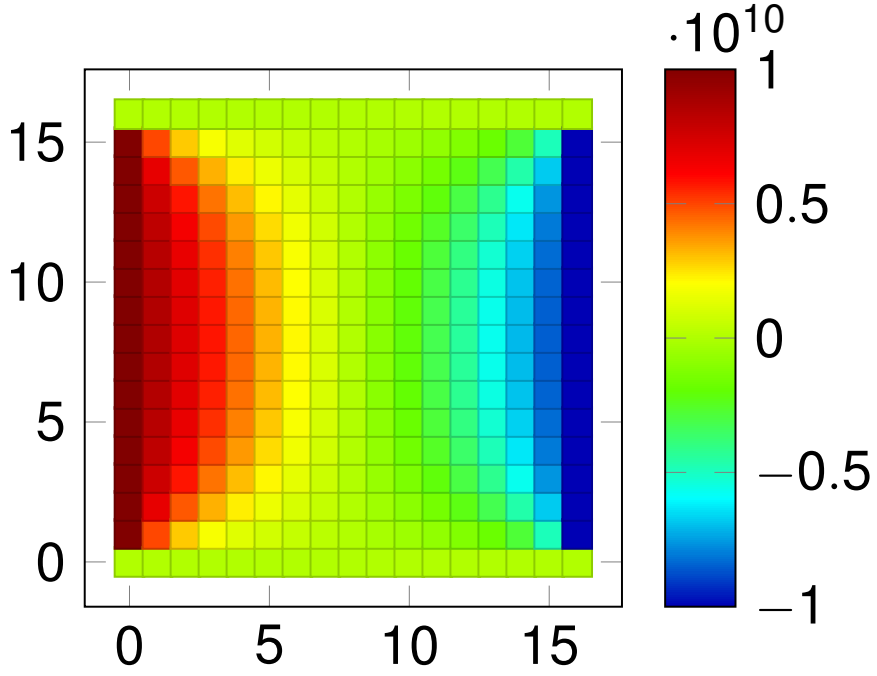
Converge if F is contracting: $d(F(x), F(y)) \leq k_F d(x, y), k_F < 1$

3. Impact of non-determinism

Example: 2D Jacobi



initial state X^0



final state at convergence

Boundary conditions: top and bottom=0, left=10^{10}*, right=*-10^{10}

3. Impact of non-determinism

Random data order $\rightarrow$ non-deterministic results

$$x_{i,j}^{n+1} = \begin{cases} f(x_{i-1,j}^n, x_{i+1,j}^n, x_{i,j-1}^n, x_{i,j+1}^n) & \text{with probability } 1/4! \\ f(x_{i+1,j}^n, x_{i,j-1}^n, x_{i-1,j}^n, x_{i,j+1}^n) & \text{with probability } 1/4! \\ \vdots \end{cases}$$

$\Rightarrow X^{n+1} = F_{\theta_{n+1}}(X^n)$ with θ_i i.i.d.

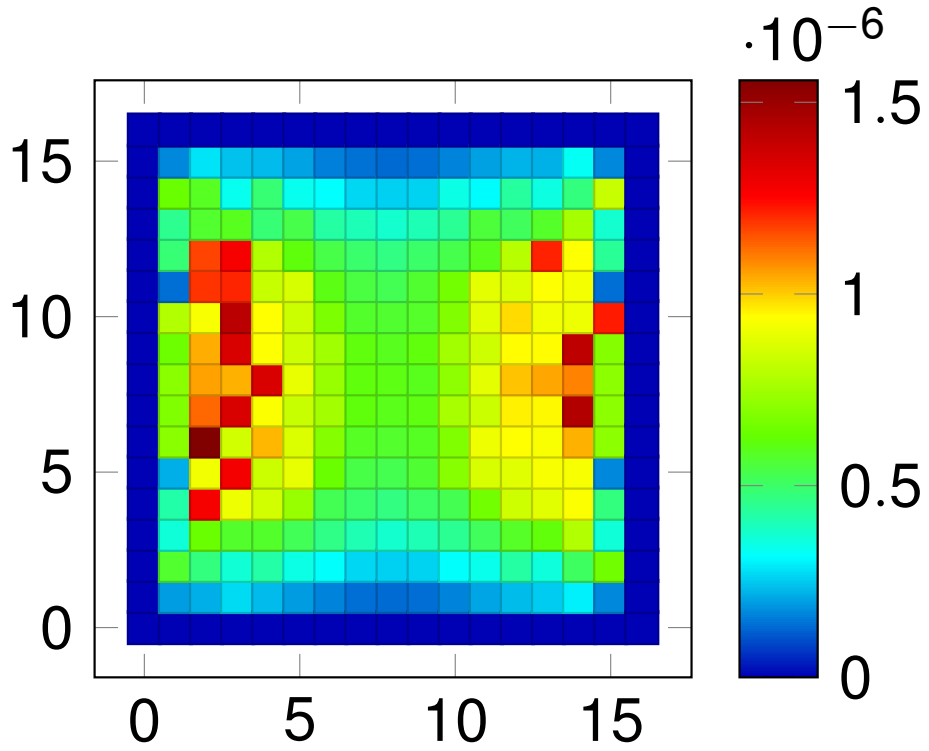
Converge (to a stationary distribution) if the F_θ functions are contracting in average: $\mathrm{E}[\log(k_{F_\theta})] < 0$

$\rightarrow$ some functions may be non-contracting if their probability is relatively low

Diaconis, P. & Freedman, D. (1999), Iterated Random Functions

3. Impact of non-determinism

Standard deviation of cell values at convergence (100 repeats):



The number of iterations also varies

Failure recovery

Failure in the (i, j) cell process $\rightarrow x_{i,j}^{n+1}$ is lost and we get back to the last checkpoint $x_{i,j}^{n-\tau}$

Iterations from $n - \tau$ to $n + 1$ must be computed again in the neighboring cells

Results may differ because of the data order randomness $\rightarrow$ iterations are restarted from a possibly different state X'^{n+1}

If X'^{n+1} remains in the same attraction domain as X^{n+1}, we will converge to the same distribution

$|x_{i,j}^{n+1} - x_{i,j}'^{n+1}|$ can be bounded thanks to round-off error bounds

Conclusion

- **Validate** large scale simulation results
- High-dimensional problems: existing **round-off error bounds** do not scale
- Different algorithms: define **code sensitivity analysis** strategies
- **Non-determinism**: delimit harmless situations

Round-off error and silent soft error propagation in exascale applications

Thank you for your attention!