Approximating a Multi-Grid Solver

Valentin Le Fèvre   Leonardo Bautista-Gomez   Marc Casas
valentin.le-fevre@ens-lyon.fr

ENS de Lyon, Barcelona Supercomputing Center

April 17, 2018

Introduction

Approximate computing: trade-off between **accuracy of result** and **execution time**.

Introduction

Approximate computing: trade-off between **accuracy of result** and **execution time**.

- Precision of a floating-point value
- No exact result exists (search query...)
- ...

Introduction

Approximate computing: trade-off between **accuracy of result** and **execution time**.

- Precision of a floating-point value
- No exact result exists (search query...)
- ...

- Skip steps in loops
- Branching to avoid useless computations
- Faulty hardware (fast adders...)
- ...

Introduction

- Multi-Grid (MG) solvers [3]: iterative solvers with different level of coarseness: number of evaluation points.
  Faster than classical method and scales well.

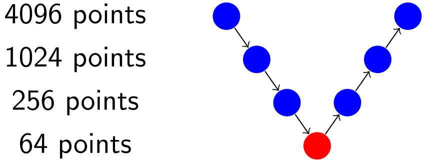


Figure: Example of cycle: each blue point represents one iteration of an iterative method, while the height corresponds to the coarseness of the grid. Red is “exact“ solve.

Introduction

- Multi-Grid (MG) solvers [3]: iterative solvers with different level of coarseness: number of evaluation points.
  Faster than classical method and scales well.

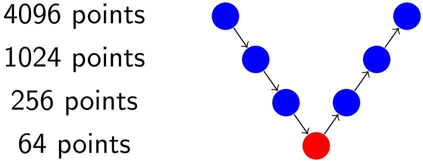


Figure: Example of cycle: each blue point represents one iteration of an iterative method, while the height corresponds to the coarseness of the grid. Red is “exact“ solve.

- Accuracy of result (*relative residual norm*) is limited by the hardware.
- We do not aim the same accuracy when using it as a conditioner or a solver.

Outline

Outline

Analysis

First idea: add more iterations at each level or more complex cycles.

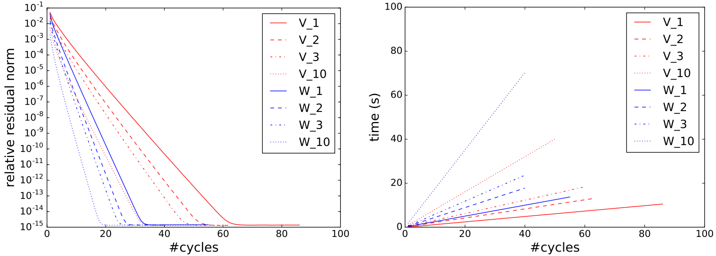


Figure: Relative residual norm and execution time as function of number of cycles.

Analysis

First idea: add more iterations at each level or more complex cycles.

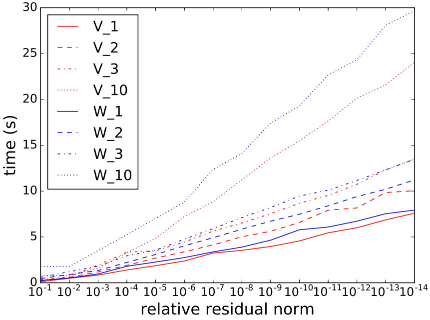


Figure: Relative residual norm as function of execution time.

Analysis

Level	Matrix size	Non-zero elements	Relax (down)	Relax (up)	Restriction	Interpolation
1	512,000	4,042,520	20 ms	20 ms	15 ms	-
2	256,000	6,475,239	20 ms	25 ms	12 ms	4 ms
3	58,893	2,000,513	8 ms	8 ms	3 ms	2 ms
4	14,285	788,509	2 ms	2 ms	1 ms	0.7 ms
5	4,238	386,333	1 ms	1 ms	0.5 ms	0.2 ms
6	609	53,493	<0.1 ms	<0.1 ms	<0.1 ms	<0.1 ms
7	69	2,873	<0.1 ms	<0.1 ms	<0.1 ms	<0.1 ms
8	2	4	<0.1 ms	-	-	<0.1 ms

Table: Time breakdown of a V-cycle with $\alpha = 1$.

$\Rightarrow$ Relaxations represent $\approx 66\%$ of the total cost of a V-cycle.

The UP-cycle

After several tries: the UP-cycle.
We do relaxations only when going up in the V-cycle.

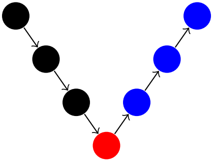
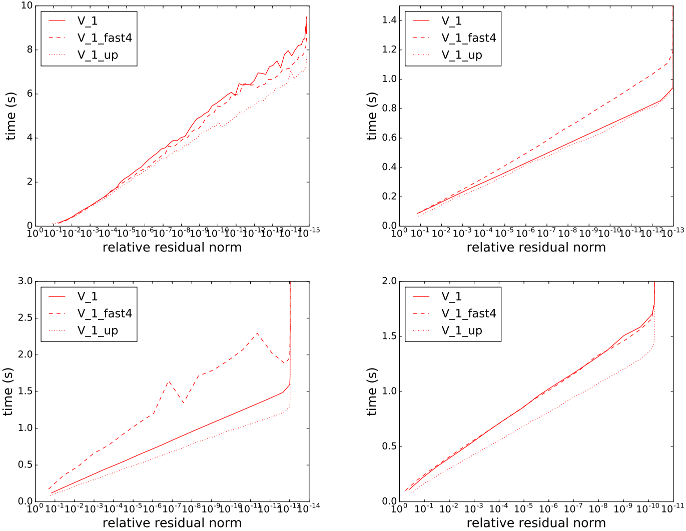
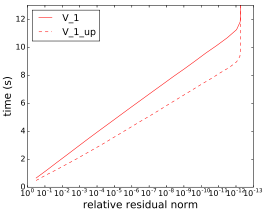


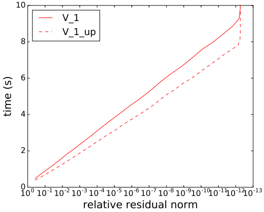
Figure: Blue: relaxation. Red: exact solve. Black: nothing.

Results

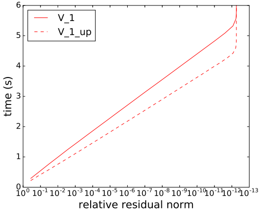




(a) 3x3x3



(b) 6x6x1



(c) 4x4x4

Overall, between 7% and 28% of improvement for reaching max accuracy on our tests.

Outline

Impact of bitwidth

- The bitwidth is a hardware limitation: we can't have results accurate to 2^{-1000} using double floating-point representation.
- However, using a small bitwidth makes computations faster and more energy-efficient [2].

Impact of bitwidth

- The bitwidth is a hardware limitation: we can't have results accurate to 2^{-1000} using double floating-point representation.
- However, using a small bitwidth makes computations faster and more energy-efficient [2].
- We rewrite the MG algorithm: one version using only single-precision floating-points and one version with the relaxation algorithm using MPFR variables (arbitrary precision) [1].

Impact of bitwidth

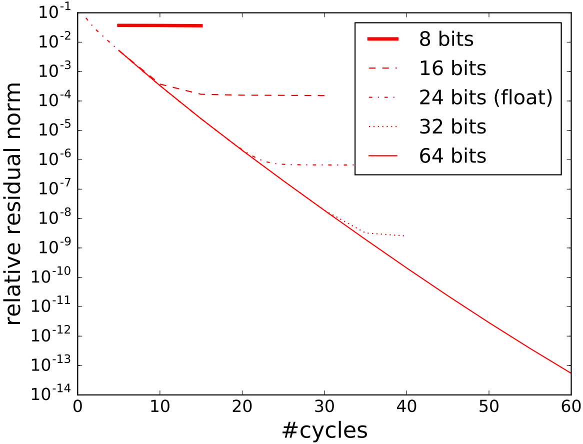


Figure: Accuracy for different number of **mantissa** bits.

Impact of bitwidth

- The bitwidth is a hardware limitation: we can't have results accurate to 2^{-1000} using double floating-point representation.
- However, using a small bitwidth makes computations faster and more energy-efficient [2].
- We rewrite the MG algorithm: one version using only single-precision floating-points and one version with the relaxation algorithm using MPFR variables (arbitrary precision) [1].

- Conclusion: using small bitwidths does not change the convergence rate (until late).

Algorithm

t a threshold, $\textsc{Update}(b)$ a function which returns an integer greater than b.

1. $b \leftarrow 64$.
2. **While** $nb\_iters < max\_iter$ **and** $rel\_res\_norm > tolerance$
    1. Do a cycle at precision b.
    2. Compute new_rel_res_norm.
    3. rel_res_norm $\leftarrow$ new_rel_res_norm.
    4. nb_iters $\leftarrow$ nb_iters$+1$.

Algorithm

t a threshold, $\textsc{Update}(b)$ a function which returns an integer greater than b.

1. $b \leftarrow 16$.
2. **While** $\mathit{nb\_iters} < \mathit{max\_iter}$ **and** $\mathit{rel\_res\_norm} > \mathit{tolerance}$
    1. Do a cycle at precision b.
    2. Compute new_rel_res_norm.
    3. **If** $\mathit{new\_rel\_res\_norm} > t \times \mathit{rel\_res\_norm}$ **Then** $b \leftarrow \textsc{Update}(b)$.
    4. rel_res_norm $\leftarrow$ new_rel_res_norm.
    5. nb_iters $\leftarrow$ nb_iters$+1$.

Algorithm

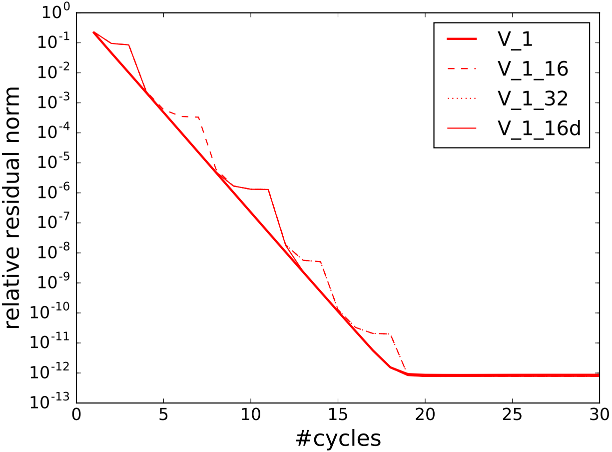


Figure: Accuracy of adaptive algorithms compared to the original double-precision with a precision threshold of 0.8.

Model

How to estimate the benefits in term of execution time ?

Model

How to estimate the benefits in term of execution time ?

$$Time(n, b) = a \cdot n^3 \cdot b^\alpha + c$$

- n: size of the problem (we worked on 3D grids so n^3 for the size of the matrix).
- b: number of mantissa bits.
- a, α, c: constants to determine.

We found $\alpha \approx 0.3$ (sublinear...) using single-precision and double-precision codes.

Results

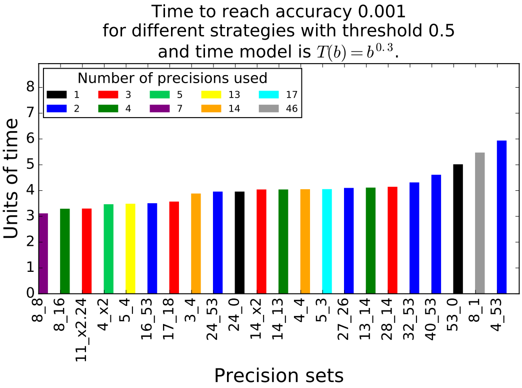


Figure: Cost of the MG solver considering several different dynamic precision scenarios to reach accuracy 10^{-3}.

GPU compared to double-precision: 34% improvement.

Results

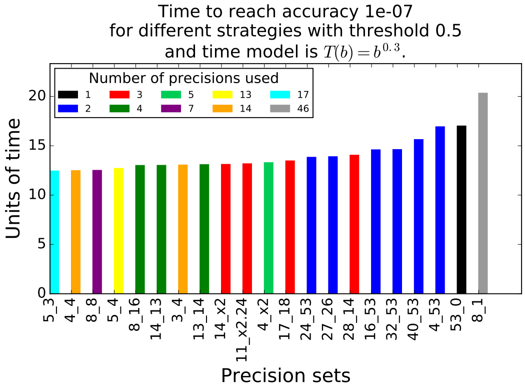


Figure: Cost of the MG solver considering several different dynamic precision scenarios to reach accuracy 10^{-7}.

GPU compared to double-precision: 23% improvement.

Results

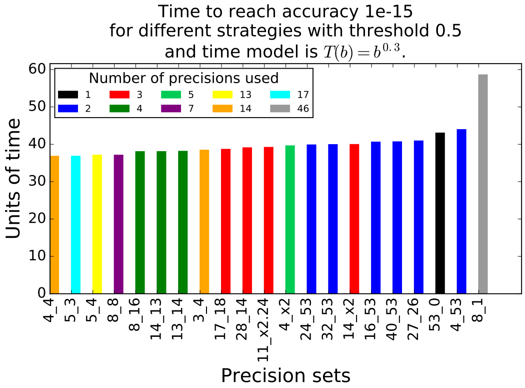


Figure: Cost of the MG solver considering several different dynamic precision scenarios to reach accuracy 10^{-15}.

GPU compared to double-precision: 9% improvement.

Outline

Conclusion (1)

Final results:

- A new cycle shape that tends to converge faster: the Up-cycle.
- A new algorithm for *any* MG cycle shape that reduces execution time and energy consumption.
- Up to 30% expected improvement on a GPU with half-precision/single-precision/double-precision available by mixing Up-cycle and changing bitwidths.
- At least 15% expected improvement for reaching maximum accuracy compared to previously.

Conclusion (2)

Future ideas:

- Model (or measure) the gains in energy consumption instead of execution time.
- Change precision **inside** a cycle?
- Link to silent data corruption: what if the environment forces us to work at 10^{-x} as max accuracy because of bitflips?

Conclusion (2)

Future ideas:

- Model (or measure) the gains in energy consumption instead of execution time.
- Change precision **inside** a cycle?
- Link to silent data corruption: what if the environment forces us to work at 10^{-x} as max accuracy because of bitflips?

Thank you for your attention. Any question?

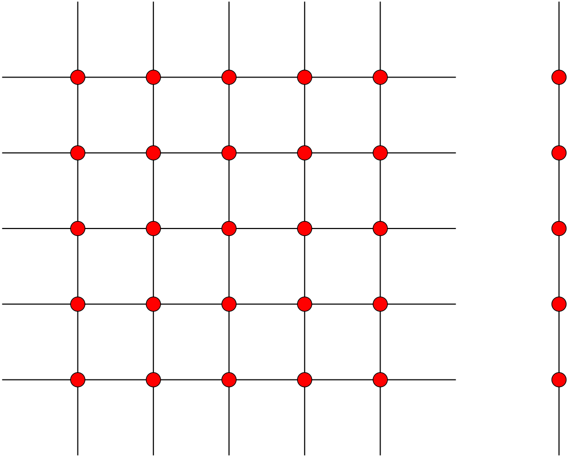
References

Laurent Fousse, Guillaume Hanrot, Vincent Lefèvre, Patrick Pélissier, and Paul Zimmermann.
Mpfr: A multiple-precision binary floating-point library with correct rounding.
ACM Trans. Math. Softw., 33(2), June 2007.

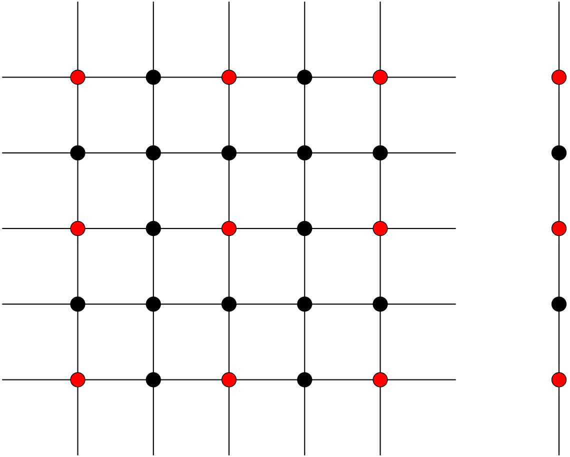
Gokul Govindu, Ling Zhuo, Seonil Choi, Padma Gundala, and Viktor K Prasanna.
Area and power performance analysis of a floating-point based application on FPGAs.
In *Seventh Annual Workshop on High Performance Embedded Computing*.

W. Hackbusch.
Multi-Grid Algorithms, pages 133–160.
Springer Berlin Heidelberg, Berlin, Heidelberg, 1991.

Grids



Grids



Grids