Efficient Floating Point Precision Tuning for Approximate Computing

POLITECNICO
MILANO 1863

Edited by: Antonio Paladini

Approximate Computing

POLITECNICO
MILANO 1863

Approximate computing

Is a computation which returns a possibly inaccurate result rather than a guaranteed accurate result, for a situation where an approximate result is sufficient for a purpose.

Approximate storage

Instead of storing data values exactly, they can be stored approximately, e.g., by truncating the lower-bits in floating point data.

Goal

Reduce the **amount of memory** needed to **run a program *P*** in an approximate version which doesn't exceed a certain **error threshold *δ*.**

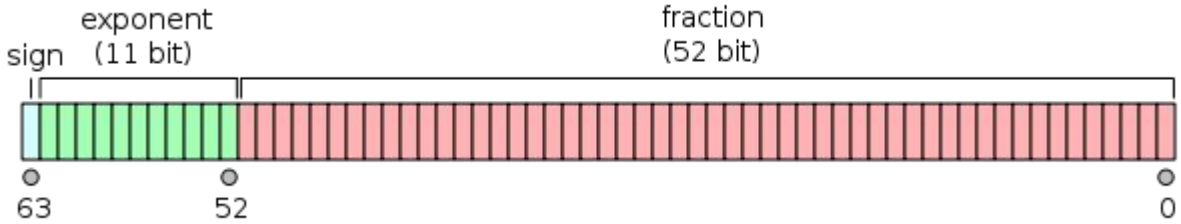
Idea: consider **arbitrary precision** floating point variables instead of common **double** (*64bit*) and **float** (*32bit*) types.

Have a look at the [GNU MPFR Library]

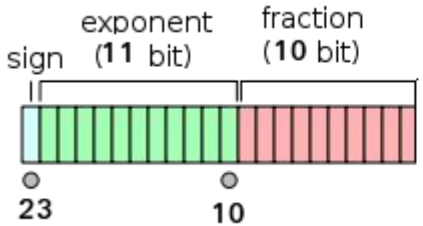
Multiple Precision Floating-Point Reliable

POLITECNICO
MILANO 1863

MPFR library allow us to set arbitrarily the significand precision of a floating-point variable.

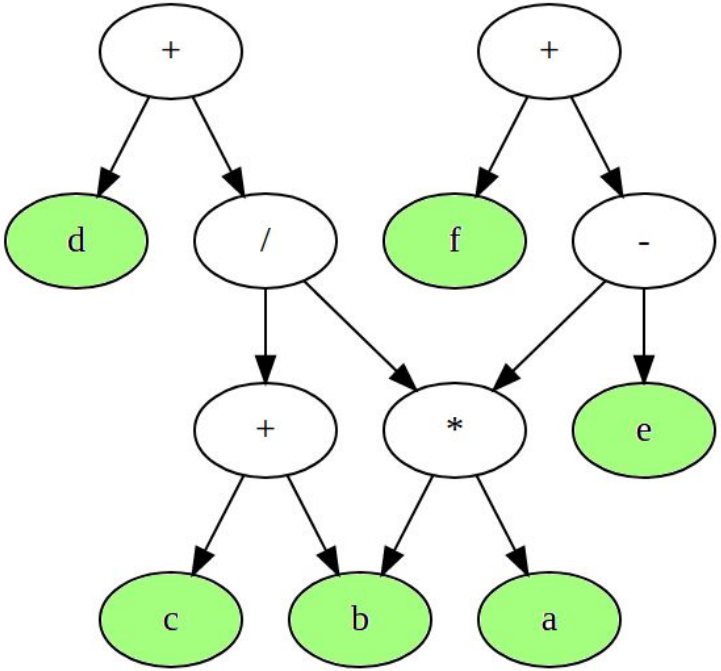


Double representation (*64bit*)
235.7239533



MPFR representation *(24 bit)*
235.5000000

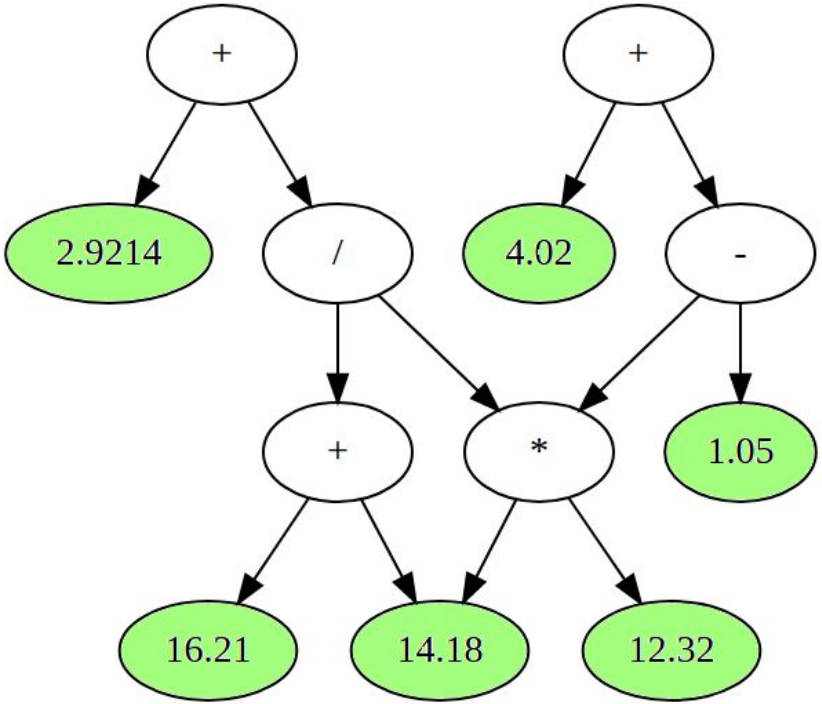
The Algorithm – Input



The input of the algorithm is a generic **Data Flow Graph**.

The graph can represent multiple expressions which are computed through **Symbolic Execution.**

The Algorithm – Compute graph

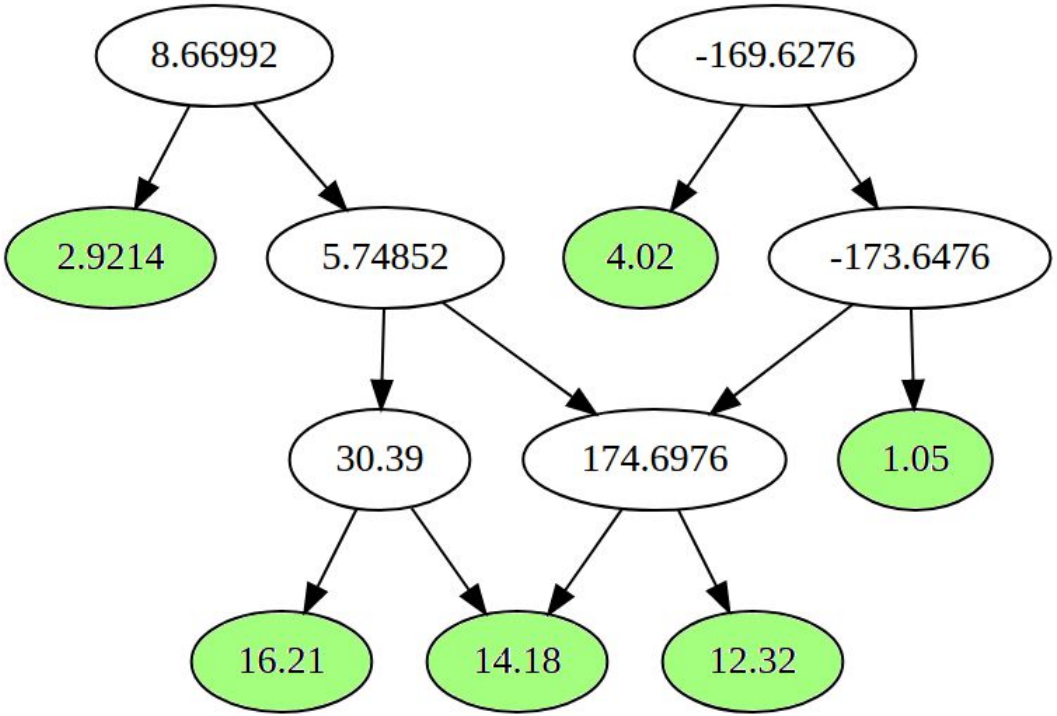


Each vertex can be either an **input** or an **operator**.

The graph is traversed through an implementation of **DFS**.

For each operand vertex visited the algorithm compute recursively the values of the operands.

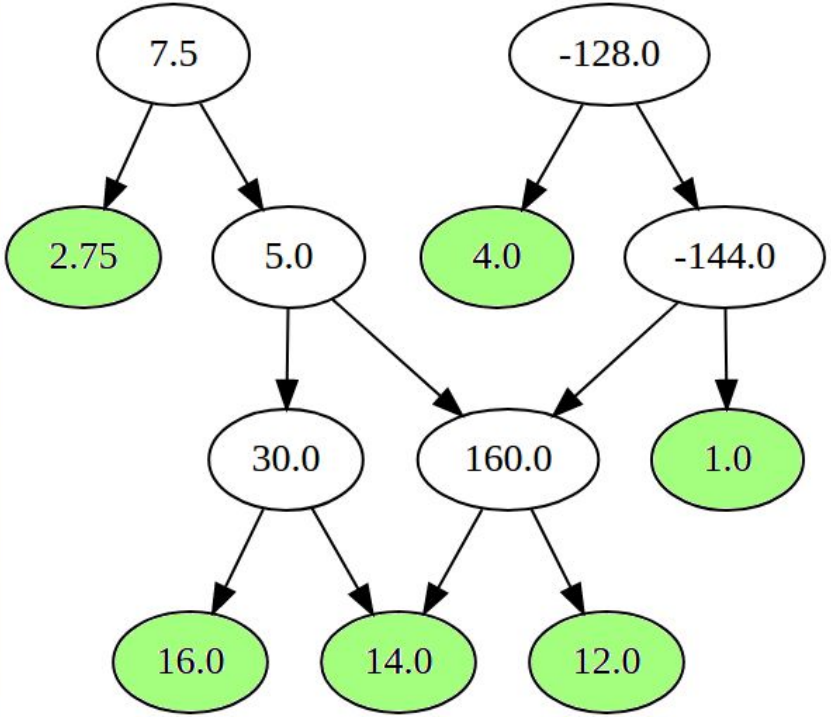
The Algorithm – Compute graph



Values obtained in this way represent the correct output of the expressions computed using double precision type (*53bit significand precision*).

Let's try to do it with something less than 53 bit.

The Algorithm – Approximation



This graph is the same as before but was computed with *4 bit* significand precision for each variable.

The error δ of this run is computed as follows:

$$SQNR = \sum_{v}^{V} \frac{Signal_v^2}{(Signal_v - Approx_v)^2}$$

$$\delta = \frac{1}{SQNR} = 0.0310575$$

The Algorithm – Threshold

POLITECNICO
MILANO 1863

Now let's set a generic threshold ***th*** .
We want to find a precision configuration ***P*** such that ***δ(P)<th.***
This will be something in the middle between:

Max Precision: P={53,53,53,53,53,53,53,53,53,53,53,53}

and

Min Precision: P={ 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4}

The Algorithm – Iterative Search

POLITECNICO
MILANO 1863

Algorithm 1 Heuristic precision tuning

1: **procedure** ITERATIVE SEARCH ▷ *main procedure*
2: $\text{MWL}_0 \leftarrow \{L_1, L_2, \ldots, L_N\}$ ▷ *initialize*
3: $P_0 \leftarrow \{U_1, U_2, \ldots, U_N\}$ ▷ *initialize*
4: **repeat**
5: $\text{MWL}_k \leftarrow \text{ISOLATED DOWNWARD}(\text{MWL}_{k-1}, P_{k-1})$
6: $P_k \leftarrow \text{GROUPED UPWARD}(\text{MWL}_k)$
7: **until** Converged
8: **return** P_k
9: **end procedure**

The Algorithm – Minimum Word Length

POLITECNICO MILANO 1863

For each variable find the minimum precision value for which, keeping the rest of the precisions unchanged, the error stay under the threshold (***δ(P)<th***).

P={53,53,53,53,53,53,53,53,53,53,53,53}	MWL={}
P={17,53,53,53,53,53,53,53,53,53,53,53}	MWL={17}
P={53,16,53,53,53,53,53,53,53,53,53,53}	MWL={17,16}
P={53,53,9,53,53,53,53,53,53,53,53,53}	MWL={17,16,9}
…	…
P={53,53,53,53,53,53,53,53,53,53,53,17}	MWL={17,16,9,8,5,10,17,9,10,9,16,17}

The Algorithm – Minimum World Length

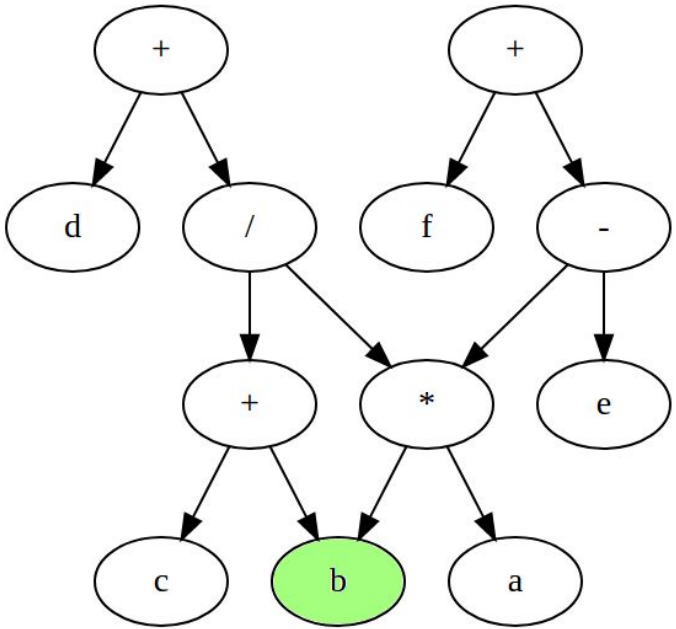
POLITECNICO MILANO 1863

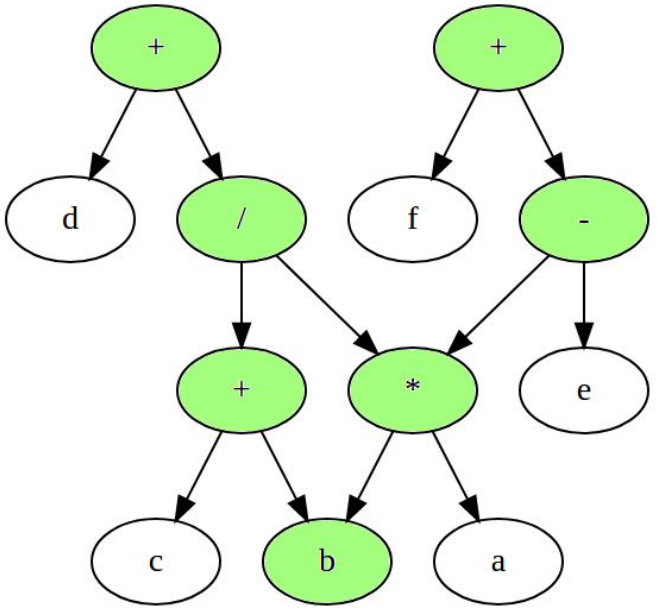
procedure ISOLATED DOWNWARD(MWL, P)
    $P_{\text{temp}} \leftarrow P$
    for $i \leftarrow 1, N$ **do** ▷ *MPI_Parallel*
        $P_{temp}[i] \leftarrow$ BINARYSEARCH($MWL[i], P[i], P, i$)
    end for
    return P_{temp}
end procedure

The Algorithm – Influence Group

POLITECNICO
MILANO 1863

Consider the **dependency graph** of a program, the **influence group G[i]** is the list of variables along some program path from x_i to the last variable that is affected by the value of x_i.





The Algorithm – Grouped Upward

POLITECNICO MILANO 1863

Increase the most convenient influence group of 1 bit for each variable until the error is under the threshold (***δ(P)<th***).

P=MWL= {17,16,9,8,5,10,17,9,10,9,16,17}	***δ(P)>th***	increased path: 1 →
P={17,17,9,8,5,10,18,10,11,10,17,18}	***δ(P)>th***	increased path: 1 →
P={17,18,9,8,5,10,19,11,12,11,18,19}	***δ(P)>th***	increased path: 1 →
P={17,19,9,8,5,10,20,12,13,12,19,20}	***δ(P)>th***	increased path: 0 →
P={18,19,9,8,5,10,21,12,14,13,20,21}	***δ(P)<th stop!***	

The Algorithm – Grouped Upward

POLITECNICO
MILANO 1863

17: ***procedure*** GROUPED UPWARD(P)
18: $\delta_{min} \leftarrow F(P)$
19: $\Delta \leftarrow \{0, 0, \ldots, 0\}$ ▷ *results from parallel threads*
20: $P_{min} \leftarrow P$
21: ***repeat***
22: ***for*** $i \leftarrow 1, N$ ***do*** ▷ MPI_Parallel
23: $P_{\text{temp}} \leftarrow \text{INCGROUPPREC}(P_{\text{min}}, i)$
24: $\Delta[i] \leftarrow F(P_{\text{temp}})$
25: **end for**
26: $\delta_{\text{min}}, I_{\text{min}} \leftarrow$ min value and its index in Δ
27: $P_{\text{min}} \leftarrow \text{INCGROUPPREC}(P_{\text{min}}, I_{\text{min}})$
28: **until** $\delta_{\text{min}} \leq \epsilon$
29: **return** P_{min}
30: **end procedure**

The Algorithm – Iterative Search

POLITECNICO
MILANO 1863

Algorithm 1 Heuristic precision tuning

1: **procedure** ITERATIVE SEARCH ▷ *main procedure*
2: $\text{MWL}_0 \leftarrow \{L_1, L_2, \ldots, L_N\}$ ▷ *initialize*
3: $P_0 \leftarrow \{U_1, U_2, \ldots, U_N\}$ ▷ *initialize*
4: **repeat**
5: $\text{MWL}_k \leftarrow \text{ISOLATED DOWNWARD}(\text{MWL}_{k-1}, P_{k-1})$
6: $P_k \leftarrow \text{GROUPED UPWARD}(\text{MWL}_k)$
7: **until** Converged
8: **return** P_k
9: **end procedure**

The Algorithm – Convergence

POLITECNICO MILANO 1863

Isolated downward and grouped upward continue to iterate until one of the following convergence criteria is satisfied:

-$MWL_k = P_k$

-$Sum\_bit(P_k) = Sum\_bit(P_{k-1})$

P={18,17,9,8,5,10,20,12,11,10,17,19}

Values of convergence

The Algorithm – Refinement

POLITECNICO
MILANO 1863

The solution found with the search algorithm presented is **strictly related to the input** that we provide to the graph. If this input changes, also the precision tuning will have different results.

A reasonable solution to this problem is to run the iterative search procedure for a training set of **randomized input** and refine final precision configuration until it satisfy the condition $\delta(P) < th$ for the 95% of the training set.

The Algorithm – Refinement

POLITECNICO MILANO 1863

Algorithm 2 Average SQNR refinement

1: **procedure** AVERAGE REFINEMENT(M)
2: $Worst\_seed \leftarrow Random\_value\_in[0, 1, \ldots, M-1]$
3: $P_{\text{refined}} \leftarrow [0, 0, \ldots, 0]$ ▷ *assign 0 for each variable*
4: **repeat**
5: $Program_k \leftarrow Program(Worst\_seed)$
6: $P_{\text{temp}} \leftarrow IterativeSearch$ on $Program_k$
7: **for** $i \leftarrow 0, Length(P_{\text{temp}})$ **do**
8: $P_{\text{refined}}[i] \leftarrow Max(P_{\text{refined}}[i], P_{\text{temp}}[i])$
9: **end for**
10: **for** $j \leftarrow 0, M-1$ **do** ▷ *MPI_Parallel*
11: *Run Program(j), Record* $SQNR_j$
12: ***end for***
13: *Find* $Worst\_seed \leftarrow$ *j causes lowest SQNR*
14: ***until*** $Average(SQNR_j) \geq Expected\_Avg$
15: ***return*** $P_{refined}$
16: ***end procedure***

Results

POLITECNICO MILANO 1863

Starting precision configuration:	P={53,53,53,53,53,53,53,53,53,53,53}
After one iteration of iterative search:	P={18,19,9,8,5,10,21,12,14,13,20,21}
Iterative Search finished:	P={18,17,9,8,5,10,20,12,11,10,17,19}
After Refinement Procedure:	P={20,19,18,17,17,18,20,19,15,21,19,19}

Metric to evaluate results is computed by summing up bits saved with the approximate version w.r.t the full precision one.

$$Memorysaving = \frac{Fullsize - Approxsize}{Fullsize}$$

It is possible to represent this program with a **memory saving of 65.09%** with an error lower than **1e-10** in the **95%** of cases.

Analysis

POLITECNICO
MILANO 1863

Input graph N vertexes	Threshold	Memory Saving
90	1e-1	**85.99%**
90	1e-5	**74,31%**
90	1e-15	**52,81%**
90	1e-25	**26,66%**

Analysis

POLITECNICO
MILANO 1863

Input graph N vertexes	Threshold	Training_set_size	N_iteration_refinement
12	1e-10	1000	**6**
20	1e-10	1000	**3**
50	1e-10	1000	**10**
90	1e-10	1000	**12**

Future work

POLITECNICO
MILANO 1863

Move from **Data Flow Graph** analysis to **Control Flow Graph** analysis to see how data approximation influence the control flow of a program.

POLITECNICO
MILANO 1863

Thank you for the attention!

Any question?