



HAL
open science

Error-free Tables for Trigonometric Function Evaluation

Hugues de Lassus Saint-Geniès, David Defour, Guillaume Revy

► **To cite this version:**

Hugues de Lassus Saint-Geniès, David Defour, Guillaume Revy. Error-free Tables for Trigonometric Function Evaluation. ARCHI: Architecture des systèmes matériels et logiciels embarqués, et méthodes de conception associées, Jun 2015, Lille, France. , 8e édition de l'école thématique Archi, 2015. lirmm-01273490

HAL Id: lirmm-01273490

<https://hal-lirmm.ccsd.cnrs.fr/lirmm-01273490>

Submitted on 12 Feb 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Error-free Tables for Trigonometric Function Evaluation

Hugues de Lassus Saint-Geniès, David Defour and Guillaume Revy

Équipe DALI – Université de Perpignan Via Domitia – LIRMM – CNRS – UMR 5506

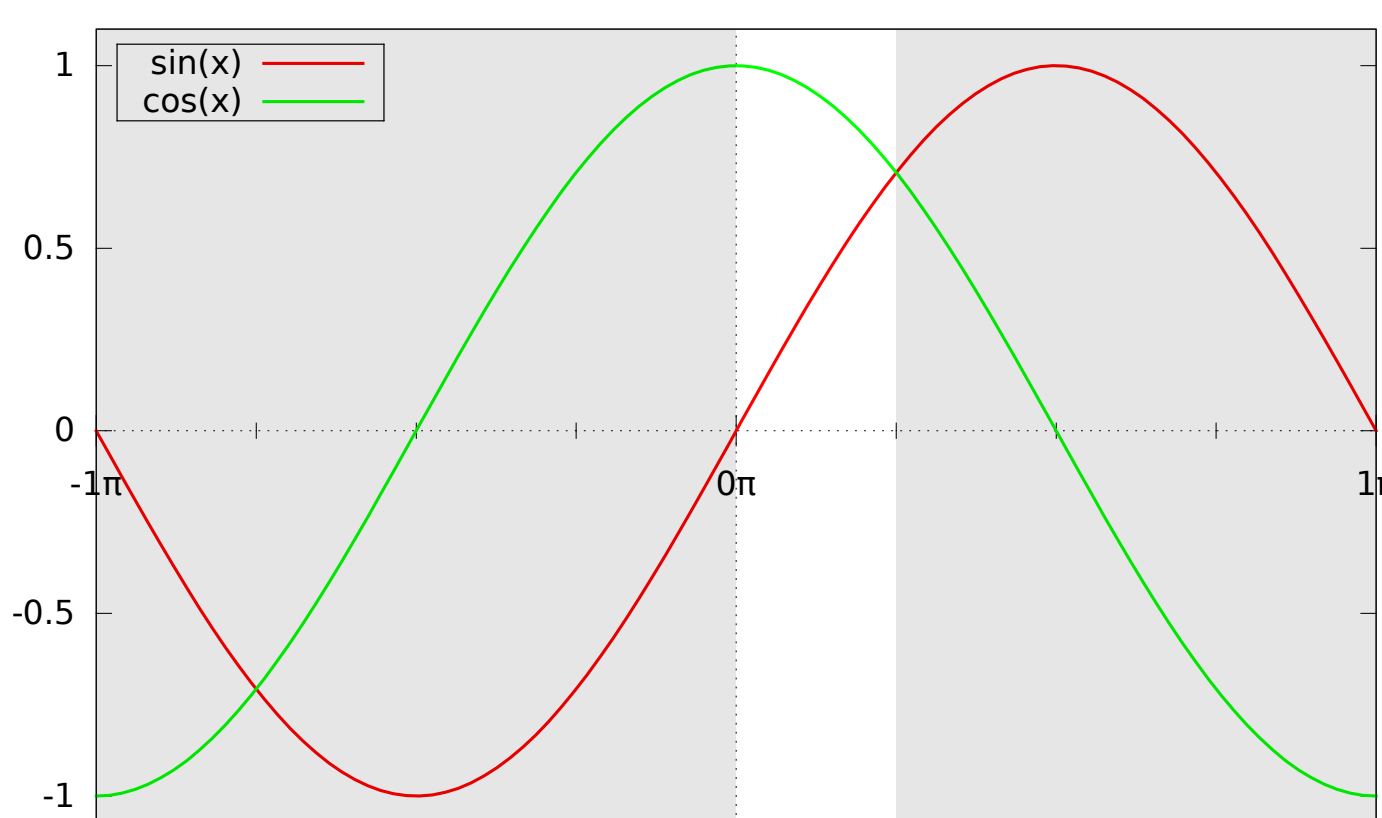


hugues.de-lassus@univ-perp.fr, david.defour@univ-perp.fr, guillaume.revy@univ-perp.fr

1. How does a modern processor calculate sine and cosine?

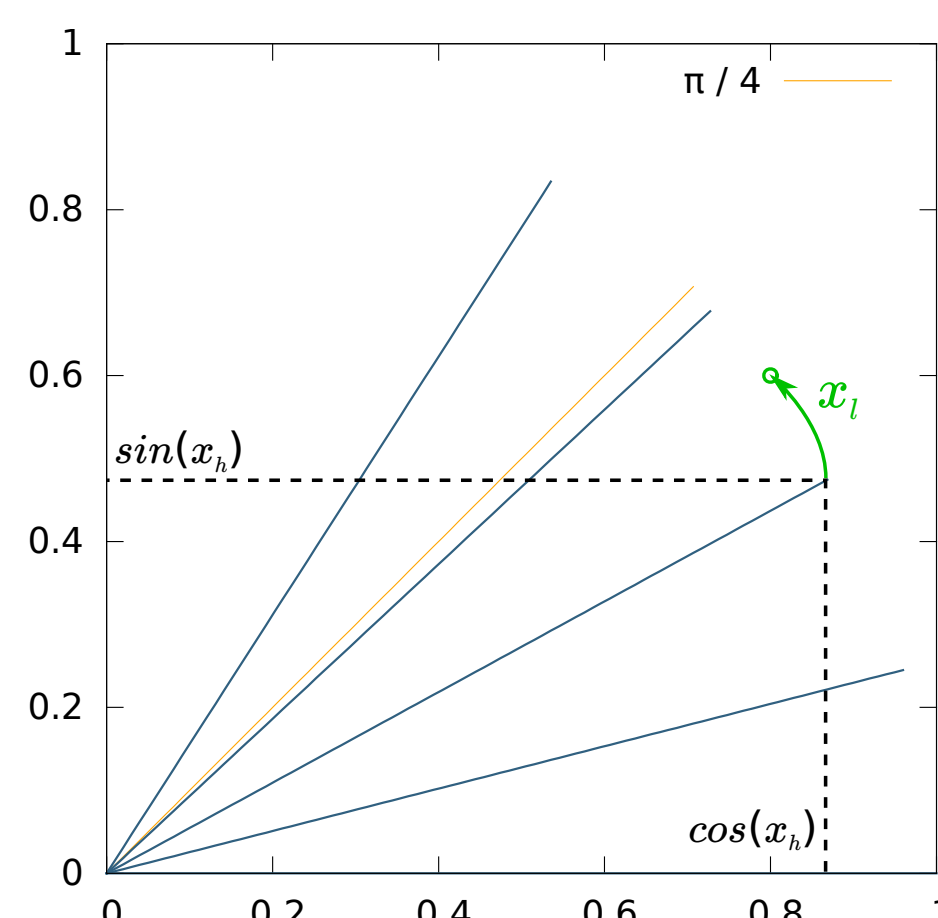
Range reduction: use of trigonometric identities

- ▶ $\sin(-x) = -\sin(x)$
- ▶ $\sin(x) = \pm f_k(x - k \cdot \frac{\pi}{2})$ with $f_k \in \{\sin, \cos\}$
- ⇒ Range reduction $\mathbb{F}_{64} \mapsto [0, \frac{\pi}{4}]$

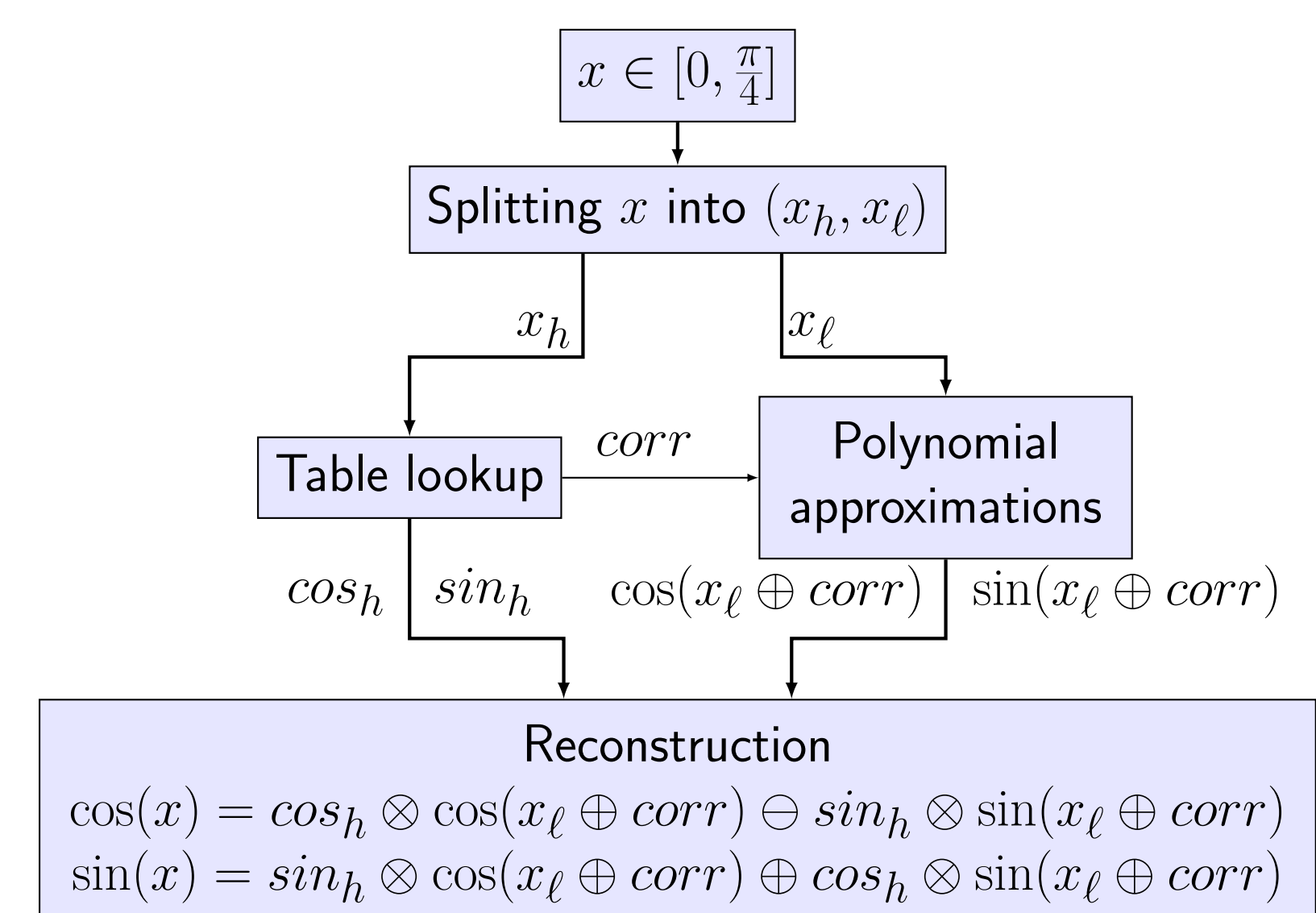


Evaluation: use of other trigonometric properties

- ▶ $\sin(x_h + x_l) = \sin(x_h) \cdot \cos(x_l) + \cos(x_h) \cdot \sin(x_l)$
- ▶ $\cos(x_h + x_l) = \cos(x_h) \cdot \cos(x_l) - \sin(x_h) \cdot \sin(x_l)$
- ▶ **Tabulated values** for sine and cosine [Tan91]

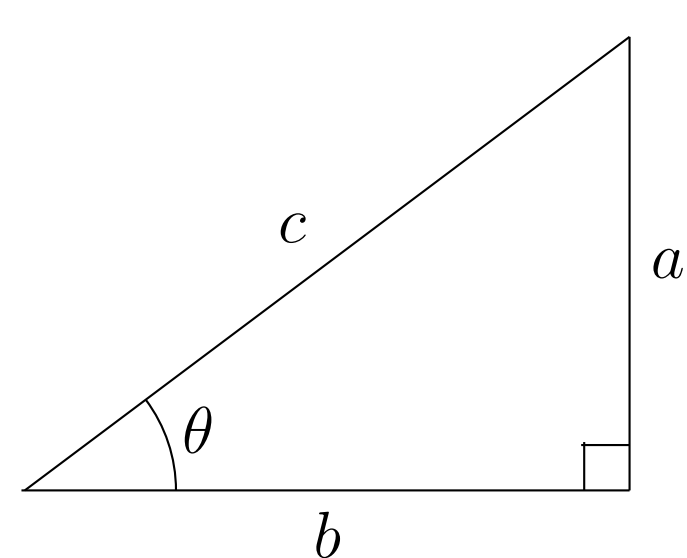


Reconstruction: scheme reducing the error on tabulated values [GB91]



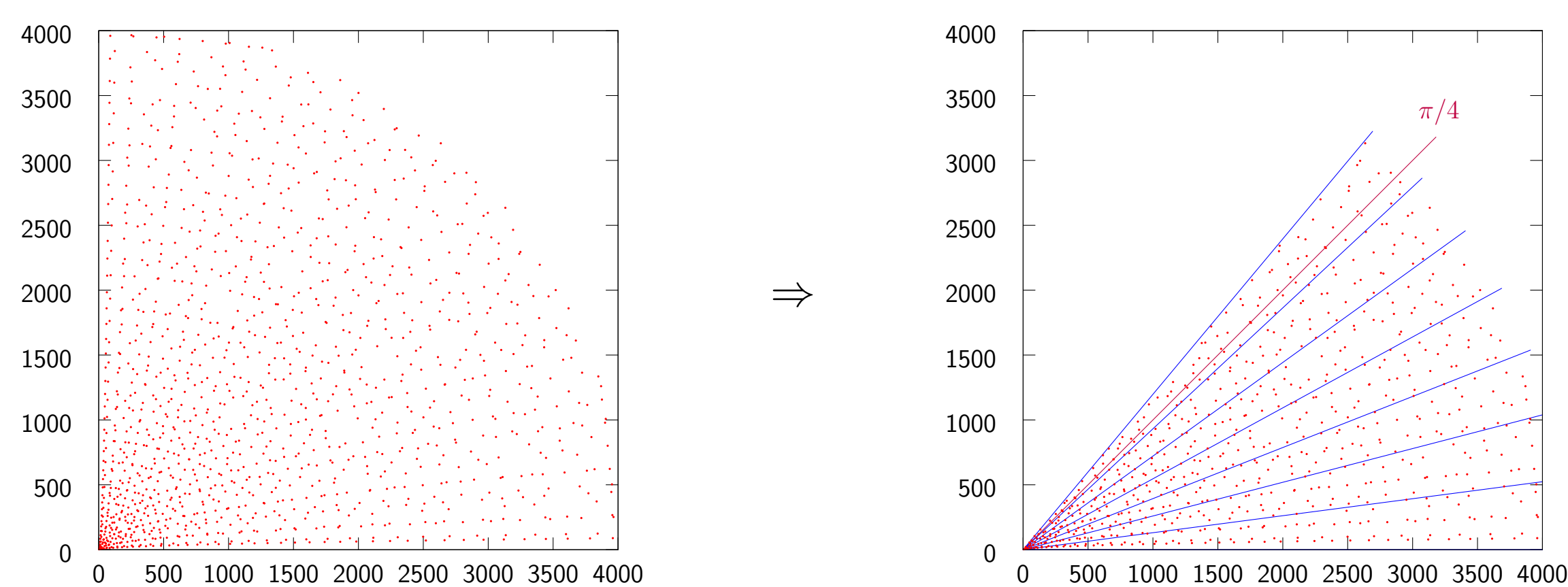
2. Pythagorean Triples

What is a Pythagorean triple?



$$\begin{cases} (a, b, c) \in \mathbb{N}^3 \\ a^2 + b^2 = c^2 \\ \sin(\theta) = \frac{a}{c} \quad \cos(\theta) = \frac{b}{c} \end{cases}$$

Primitive Pythagorean Triple: a Pythagorean triple (a, b, c) for which $\gcd(a, b, c) = 1$.



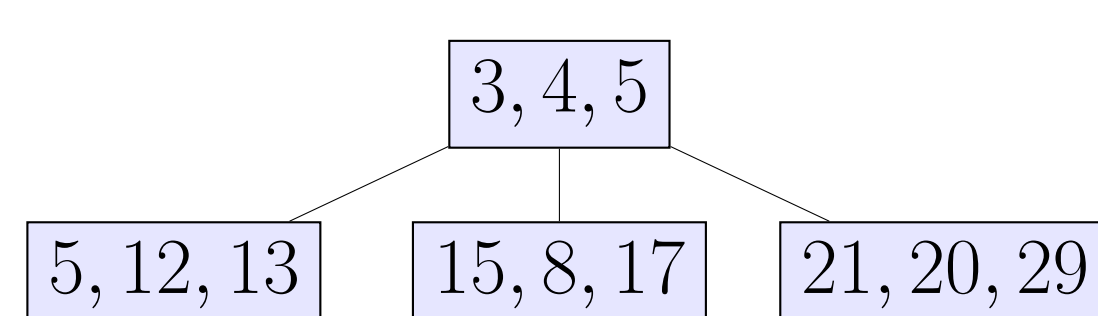
Primitive Pythagorean triples with $c \leq 2^{12}$

Only a subset fits in a table.

3. Primitive Pythagorean Triple Generation

Barning-Hall ternary-tree structure:

$$\begin{pmatrix} 1 & -2 & 2 \\ 2 & -1 & 2 \\ 2 & -2 & 3 \end{pmatrix}, \begin{pmatrix} -1 & 2 & 2 \\ 2 & -1 & 2 \\ -2 & 2 & 3 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 3 \end{pmatrix}$$



Several equivalent trees, easy to implement

Proven to generate all primitive triples by increasing hypotenuse lengths [Bar63]

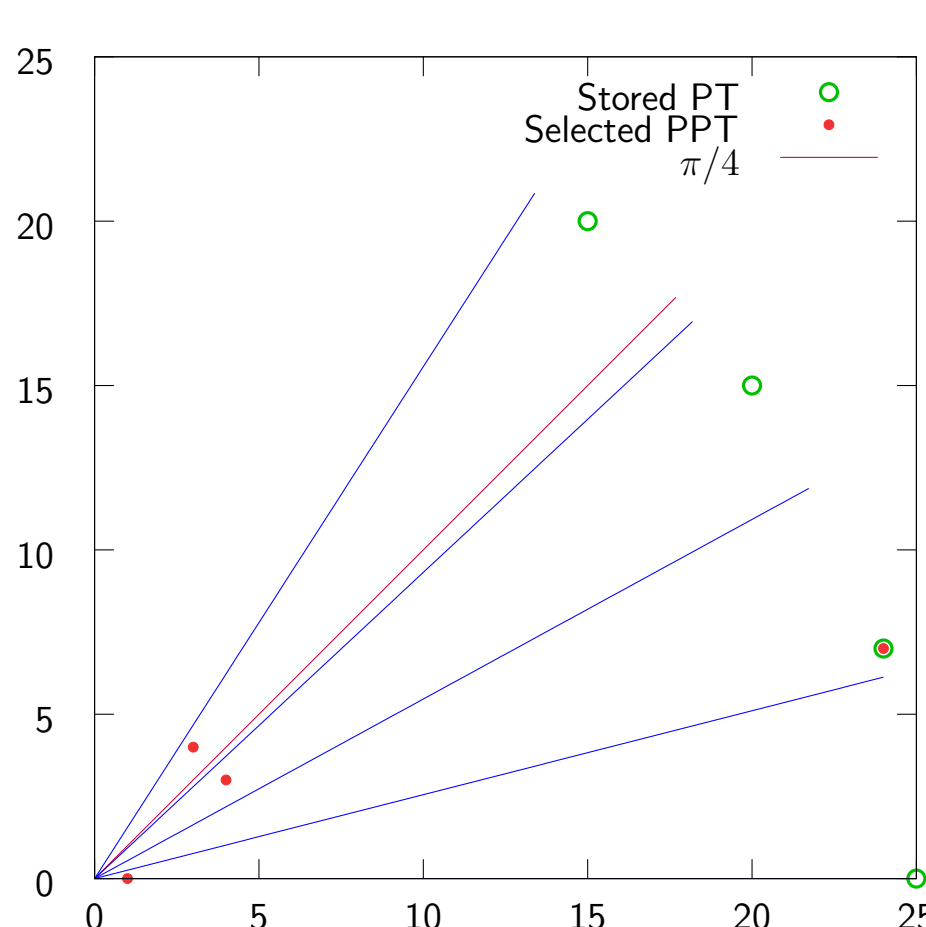
4. Primitive Pythagorean Triple Selection

Only one triple per entry needed

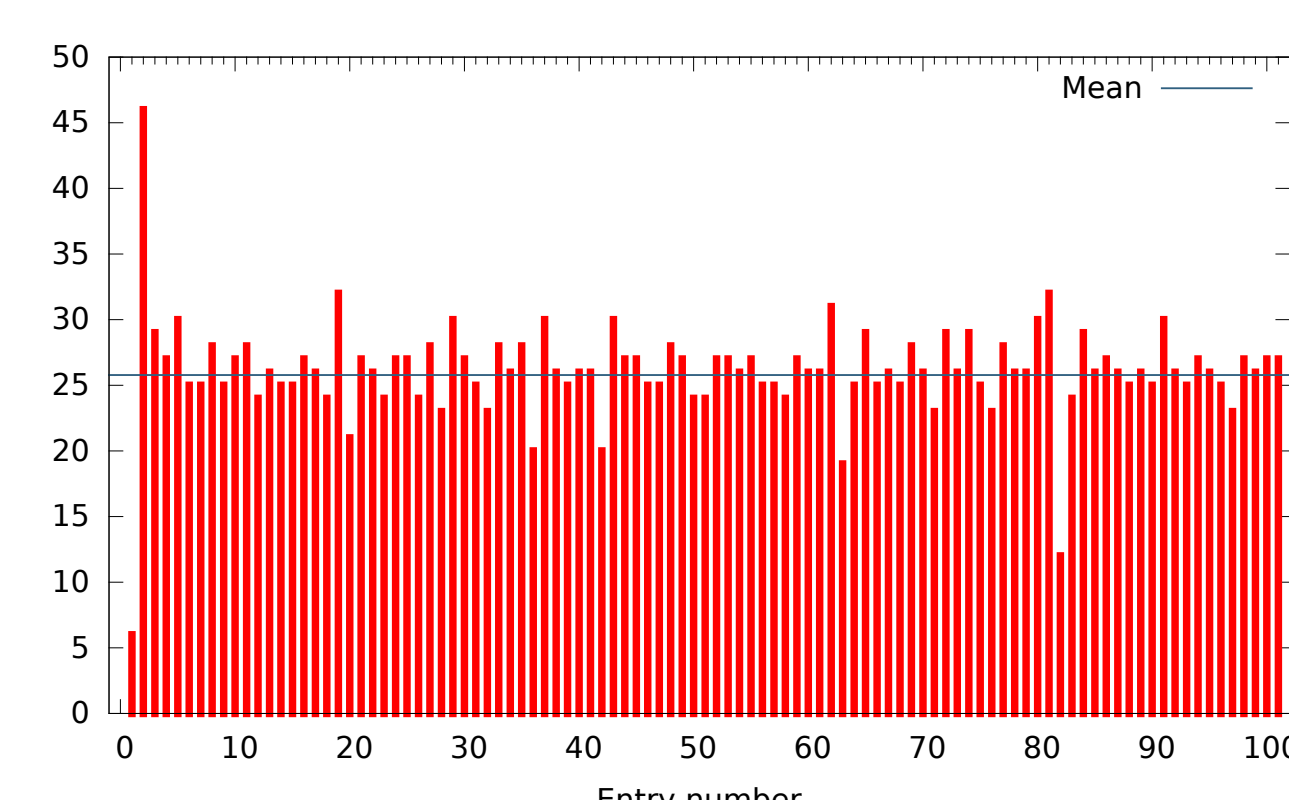
Division by c removed by incorporating it in polynomial approximations

⇒ **Same hypotenuse** c needed for all entries

⇒ **Scale** selected PPTs to the **least common multiple** of their hypotenuses



A simple example for a 2-bit indexed table



Number of PPTs/entry for a 7-bit indexed table: roughly 26^{100} combinations of 1 PPT/entry.

5. Exhaustive Search for a Small Common Hypotenuse

Algorithm

- 1: $n \leftarrow 4$
- 2: **repeat**
- 3: Generate all PPTs (a, b, c) such that $c \leq 2^n$.
- 4: Search for the LCM k among all generated hypotenuses c .
- 5: $n \leftarrow n + 1$
- 6: **until** such a k is found
- 7: Build tabulated values $(A, B, corr)$ for every entry.

Results

p	k_{min}	n	time (s)	Triples	Hypotenuses
3	425	9	$\ll 1$	86	66
4	5525	13	$\ll 1$	1404	889
5	160,225	18	0.2	42,328	24,228
6	1,698,385	21	7	335,344	179,632
7	6,569,225	23	31	1,347,953	686,701
8	$> 2^{27}$	> 27	$> 6700?$	$> 21,407,992$	$> 10,144,723$

▶ Impossible to generate tables indexed by **more than 7 bits**.

▶ 8 to 10 bit-indexed tables desired to **optimize caching**.

6. Heuristic Search

Prime factorization of found common multiples

k	Prime factorization
425	$5^2 \cdot 17$
5525	$5^2 \cdot 13 \cdot 17$
160,225	$5^2 \cdot 13 \cdot 17 \cdot 29$
1,698,385	$5 \cdot 13 \cdot 17 \cdot 29 \cdot 53$
6,569,225	$5^2 \cdot 13 \cdot 17 \cdot 29 \cdot 41$

Heuristic: store primitive Pythagorean triples satisfying

$$c = \prod_i p_i^{r_i} \quad \text{with}$$

- ▶ $r_i \in \{0, 1\}$ if $p_i \neq 5$
- ▶ $r_i \in \mathbb{N}^*$ else
- ▶ and $p_i \in \mathcal{P}$

where \mathcal{P} is the set of Pythagorean primes ≤ 73 :

$$\mathcal{P} = \{5, 13, 17, 29, 37, 41, 53, 61, 73\}$$

Results

p	k_{min}	n	time (s)	triples	hypotenuses
6	1,698,385	21	0.1	2171	66
7	6,569,225	23	0.4	3452	69
8	314,201,225	29	9.5	10,467	100
9	12,882,250,225	34	294	20,311	109
10	279,827,610,985	39	9393	33,056	110

▶ **> 99 % less memory usage**

▶ **> 99 % time saved at generation**

▶ **Same tables** for $p \in \{3, 7\}$

7. Theoretical Gains

Comparison between three table-based range reductions, for $p = 10$. The number of memory accesses (MA) and the number of floating point operations (FLOP) are reported.

Solution	Quick phase (66 bits)	Accurate phase (150 bits)	Table size (bytes)
Tang	4 MA + 64 FLOP	6 MA + 241 FLOP	38640
Gal	3 MA + 53 FLOP	9 MA + 268 FLOP	57960
Proposed	3 MA + 53 FLOP	5 MA + 148 FLOP	32200

References

- [Bar63] F. J. M. Barning. On pythagorean and quasi-pythagorean triangles and a generation process with the help of unimodular matrices. *(Dutch) Math. Centrum Amsterdam Afd. Zuivere Wisk. ZW-001*, 1963.
- [GB91] S. Gal and B. Bachelis. An accurate elementary mathematical library for the IEEE floating point standard. *ACM Trans. Math. Software*, 17:26–45, 1991.
- [Tan91] Ping Tak Peter Tang. Table-lookup algorithms for elementary functions and their error analysis. In Peter Kornerup and David W. Matula, editors, *Proceedings: 10th IEEE Symposium on Computer Arithmetic: June 26–28, 1991, Grenoble, France*, pages 232–236, pub-IEEE:adr, 1991. IEEE Computer Society Press.