Wigner 3j, 6j and 9j symbols - hashing and priming

$$\begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix} \quad \begin{Bmatrix} a & b & c \\ d & e & f \end{Bmatrix} \quad \begin{Bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{Bmatrix}$$

Fast and accurate!

Håkan T. Johansson, Chalmers, Göteborg

Christan Forssén, Jimmy Rotureau

Chalmers & UT / ORNL, TN, USA Chalmers & NSCL / MSU, MI, USA

SFS-KF XXXV, Lund, November 2015

Wigner 3j, 6j and 9j symbols

Theoreticians are known to use them:

$$u_\alpha^{A\lambda JT}(\eta,\nu) = \sum_{\substack{n_\eta l_\eta \\ n_\nu l_\nu \cdots}} \frac{R_{n_\eta l_\eta}(\eta) R_{n_\nu l_\nu}(\nu)}{\langle n_\eta l_\eta 00 l_\eta | 00 n_\eta l_\eta l_\eta \rangle_{\frac{2}{A-2}}} (-1)^{3I_1+I_{23}+J_{ab}-T_{23}-S+L} \langle n_a l_a n_b l_b L | n_\eta l_\eta n_\nu l_\nu L \rangle_1$$

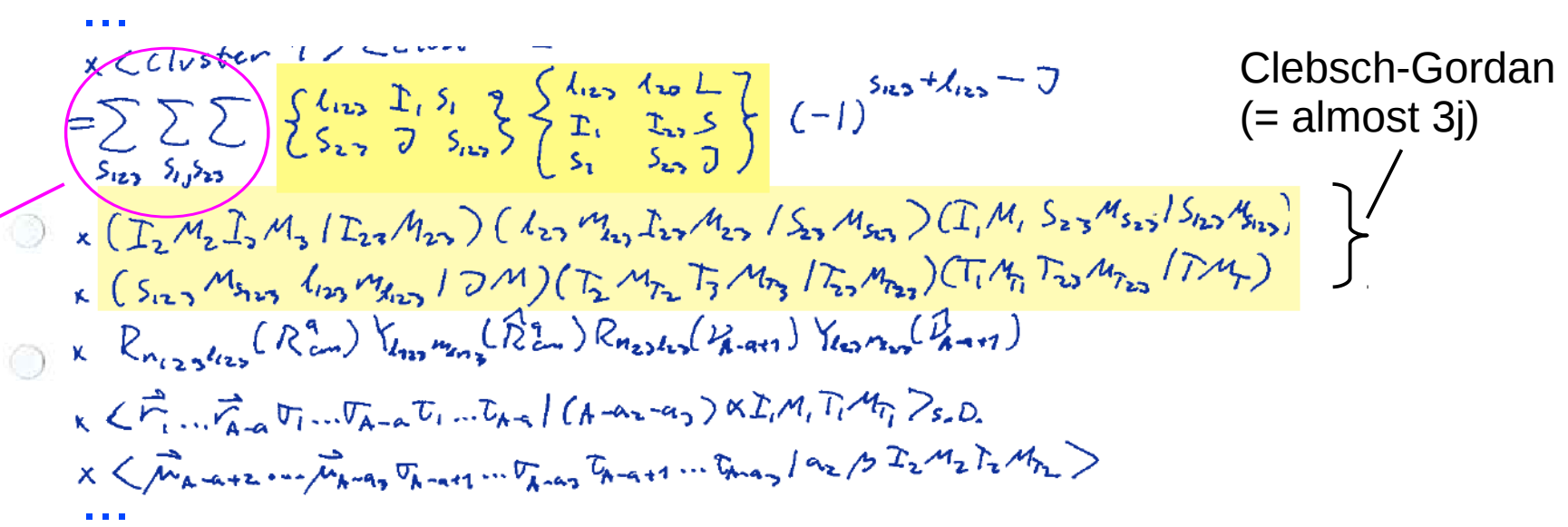
$$\times \frac{\hat{L}\hat{S}\hat{J}_{ab}^2 \hat{j}_a \hat{j}_b}{\hat{J}\hat{T}} \begin{Bmatrix} L & I_{23} & J_{ab} \\ I_1 & J & S \end{Bmatrix} \begin{Bmatrix} l_a & l_b & L \\ I_3 & I_2 & I_{23} \\ j_a & j_b & J_{ab} \end{Bmatrix} {}_{\mathrm{SD}}\langle A\lambda JT \| \left[a^\dagger_{n_a l_a j_a t_a} a^\dagger_{n_b l_b j_b t_b} \right]^{J_{ab} T_{ab}} \| (A-2)\alpha_1 I_1 T_1 \rangle_{\mathrm{SD}}$$

And even like them...

[D. Sääf and C. Forssén, Phys Rev C. 89. 011303 (2014)]

A lot!

Clebsch-Gordan
(= almost 3j)



Wigner 6j formula

Factorials!

$$\left\{ \begin{array}{ccc} a & b & c \\ d & e & f \end{array} \right\} = \prod_{i=1}^{4} \sqrt{\Delta(\hat{\alpha}_i)} \sum_k (-1)^k \frac{(k+1)!}{\prod_{i=1}^{4}(k-\alpha_i)! \prod_{j=1}^{3}(\beta_j - k)!},$$

$$\Delta(a,b,c) = \frac{(a+b-c)!(a-b+c)!(b+c-a)!}{(a+b+c+1)!},$$

$$\begin{array}{lll} \hat{\alpha}_1 = (a,b,c), & \alpha_1 = a+b+c, & \beta_1 = a+b+d+e, \\ \hat{\alpha}_2 = (d,e,c), & \alpha_2 = d+e+c, & \beta_2 = a+c+d+f, \\ \hat{\alpha}_3 = (a,e,f), & \alpha_3 = a+e+f, & \beta_3 = b+c+e+f, \\ \hat{\alpha}_4 = (d,b,f), & \alpha_4 = d+b+f. & \end{array}$$

Wigner 6j formula

$$W_x = \frac{n\sqrt{s}}{q}$$

An alternating sum of fractions of products of factorials

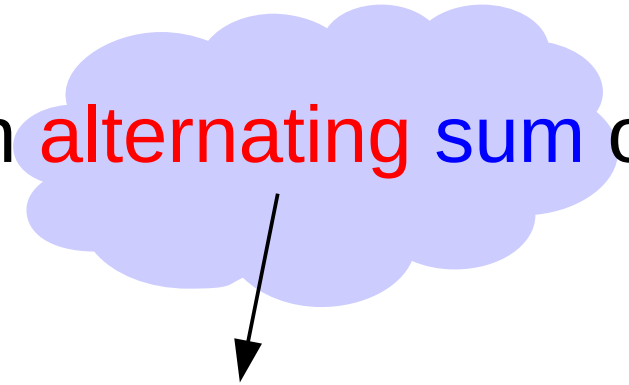
$$\left\{ \begin{array}{ccc} a & b & c \\ d & e & f \end{array} \right\} = \prod_{i=1}^{4} \sqrt{\Delta(\hat{\alpha}_i)} \sum_k (-1)^k \frac{(k+1)!}{\prod_{i=1}^{4}(k-\alpha_i)! \prod_{j=1}^{3}(\beta_j - k)!},$$

$$\Delta(a,b,c) = \frac{(a+b-c)!(a-b+c)!(b+c-a)!}{(a+b+c+1)!},$$

$$\begin{array}{lll} \hat{\alpha}_1 = (a,b,c), & \alpha_1 = a+b+c, & \beta_1 = a+b+d+e, \\ \hat{\alpha}_2 = (d,e,c), & \alpha_2 = d+e+c, & \beta_2 = a+c+d+f, \\ \hat{\alpha}_3 = (a,e,f), & \alpha_3 = a+e+f, & \beta_3 = b+c+e+f, \\ \hat{\alpha}_4 = (d,b,f), & \alpha_4 = d+b+f. & \end{array}$$

Catastrophic cancellation

An alternating sum of fractions of products of factorials



Summing terms with opposite sign in floating point → precision loss:

1.00010000 – 1.00000000 = 0.00010000xxx

1.00000010 – 1.00000000 = 0.00000010xxxxxx

1.00000001 – 1.00000000 = 0.00000001xxxxxxx

Loss of significance

Solution: more significant digits – preferably all of them?

LCD – least common denominator

An alternating sum of fractions of products of factorials

Kindergarten arithmetics: sum of fractions is done with LCD:

$$\frac{3}{7}-\frac{7}{9}=\frac{9\cdot 3-7\cdot 7}{9\cdot 7}=\frac{27-49}{63}=\frac{-22}{63}$$

→ Exact sum of numerator can be obtained using integers!

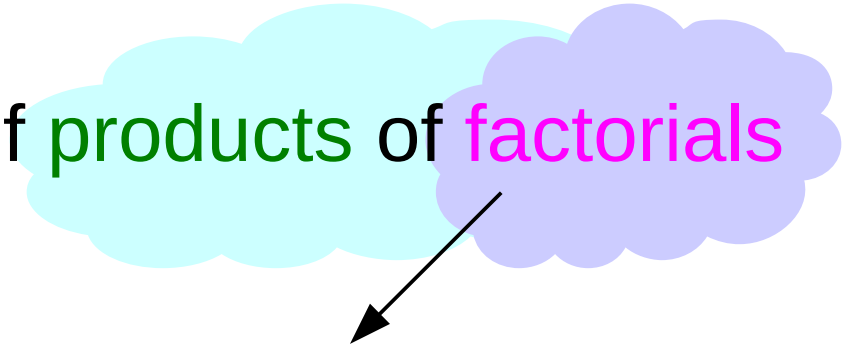
Not so kindergarten: large integers: (from { 50 50 50 ; 50 50 50 })

```
  0x0000000000000001
- 0x000000000121eac0
+ 0x00001370d64dcb8c
- 0x3e5f5a3997b68000
  ...
+ 0x0000000069d2b107 fef1734e54fb43b1 f60b61893272ca50
= 0xffffffeb979646a 8ab7d1d8439c3382 2b16af8f9ec9ad89 4af8ae3f1f3cd594
```

Multi-word integers!

Prime number factorisation

An alternating sum of fractions of products of factorials



Factorials are large:

50! = 30414093201713378043612608166064768844377641568960512000000000000

Prime factorised representation is easier:

$$50! = 2^{47} \cdot 3^{22} \cdot 5^{12} \cdot 7^{8} \cdot 11^{4} \cdot 13^{3} \cdot 17^{2} \cdot 19^{2} \cdot 23^{2} \cdot 29^{1} \cdot 31^{1} \cdot 37^{1} \cdot 41^{1} \cdot 43^{1} \cdot 47^{1}$$

Multiplication is straightforward:

$$7! \cdot 5! = 2^{4} \cdot 3^{2} \cdot 5^{1} \cdot 7^{1} \cdot 2^{3} \cdot 3^{1} \cdot 5^{1} = 2^{4+3} \cdot 3^{2+1} \cdot 5^{1+1} \cdot 7^{1+0} = 2^{7} \cdot 3^{3} \cdot 5^{2} \cdot 7^{1}$$

And division:

$$\frac{7!}{5!} = \frac{2^{4} \cdot 3^{2} \cdot 5^{1} \cdot 7^{1}}{2^{3} \cdot 3^{1} \cdot 5^{1}} = 2^{4-3} \cdot 3^{2-1} \cdot 5^{1-1} \cdot 7^{1-0} = 2^{1} \cdot 3^{1} \cdot 5^{0} \cdot 7^{1}$$

Prime number factorisation (cont'd)

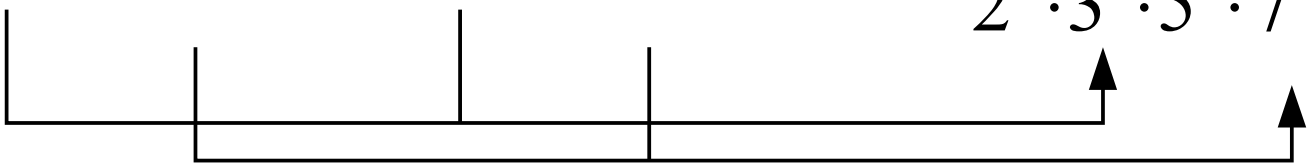
An alternating sum of fractions of products of factorials

Prime factorisation is easier:

$$50! = 2^{47} \cdot 3^{22} \cdot 5^{12} \cdot 7^{8} \cdot 11^{4} \cdot 13^{3} \cdot 17^{2} \cdot 19^{2} \cdot 23^{2} \cdot 29^{1} \cdot 31^{1} \cdot 37^{1} \cdot 41^{1} \cdot 43^{1} \cdot 47^{1}$$

Finding the LCD is finding the minimum exponents:

$$\frac{3}{7} - \frac{7}{9} = 2^{0} \cdot 3^{1} \cdot 5^{0} \cdot 7^{-1} - 2^{0} \cdot 3^{-2} \cdot 5^{0} \cdot 7^{1} = \frac{2^{0} \cdot 3^{3} \cdot 5^{0} \cdot 7^{0} - 2^{0} \cdot 3^{0} \cdot 5^{0} \cdot 7^{2}}{2^{0} \cdot 3^{2} \cdot 5^{0} \cdot 7^{1}}$$



Addition not that easy... → convert to and use (multi-word) integers

$$\ldots = \frac{27 - 49}{2^{0} \cdot 3^{2} \cdot 5^{0} \cdot 7^{1}} = \frac{-22}{2^{0} \cdot 3^{2} \cdot 5^{0} \cdot 7^{1}}$$

Wigner 6j formula – almost there!

An alternating sum of fractions of products of factorials

$$\left\{ \begin{array}{ccc} a & b & c \\ d & e & f \end{array} \right\} = \prod_{i=1}^{4} \sqrt{\Delta(\hat{\alpha}_i)} \sum_k (-1)^k \frac{(k+1)!}{\prod_{i=1}^{4}(k-\alpha_i)! \prod_{j=1}^{3}(\beta_j - k)!},$$

Prefactor: just some more factorials
(→ a few more exponents)

$$\Delta(a,b,c) = \frac{(a+b-c)!(a-b+c)!(b+c-a)!}{(a+b+c+1)!},$$

We thus can reach a form having only three (large) integers:

$$W_x = \frac{n\sqrt{s}}{q}$$

Fixed relative accuracy!

$$W_x = \frac{n\sqrt{s}}{q}$$

Final evaluation requires:

- 3 conversions from integer to floating point
- 1 square root
- 1 multiplication
- 1 division

Each introduces at most:
rounding error of *half a least significant bit* (ε_{mach}).

64-bit format of IEEE 758 (C's `double`): $\varepsilon_{mach} = 1.11 \cdot 10^{-16}$

→ `WIGXJPF` calculates any 3j/6j/9j with a fixed relative accuracy:

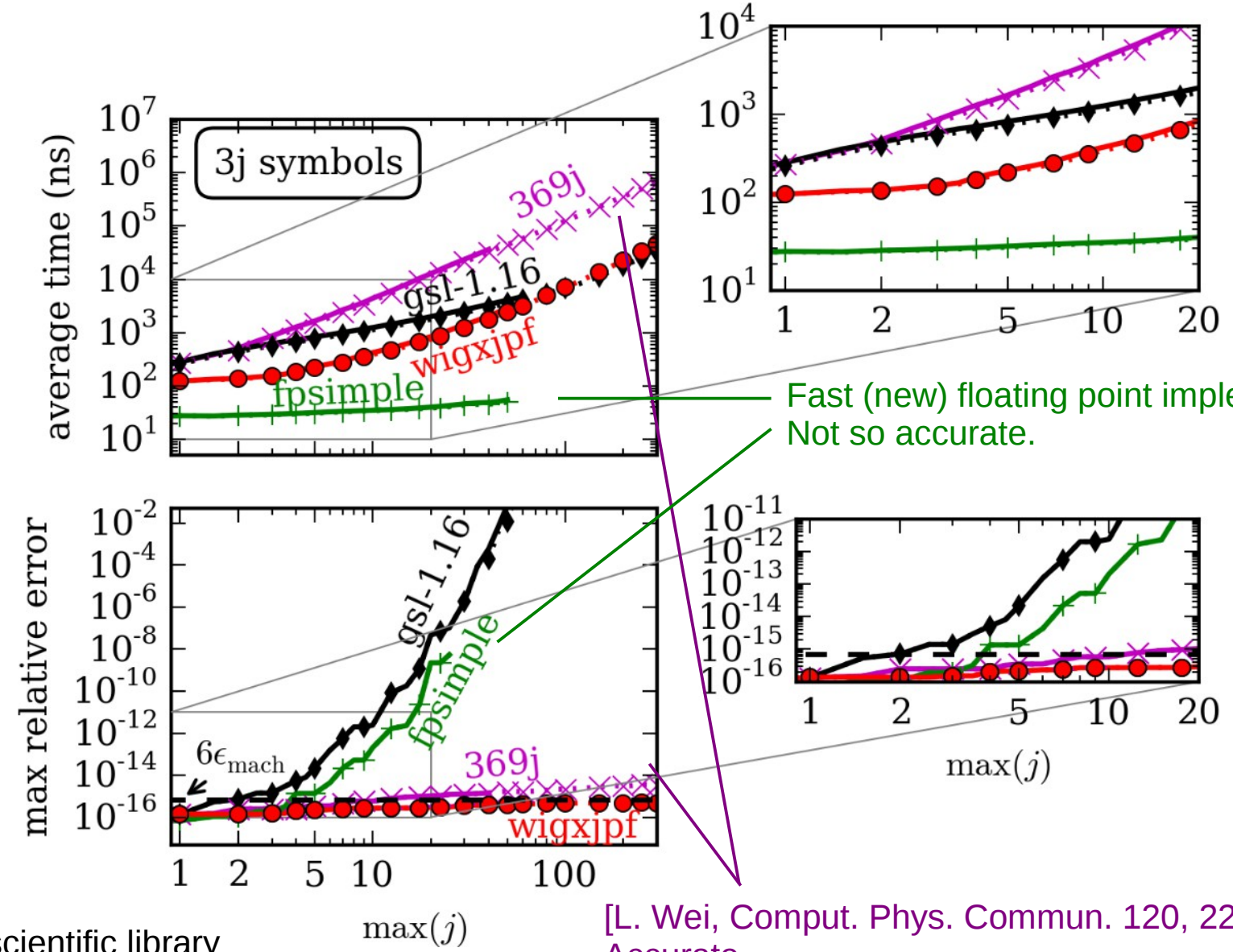
$6.66 \cdot 10^{-16}$

Some examples

TABLE I. Values and execution times for selected xj symbols. These times include setup of the precalculated tables of factorised factorials. The memory used for those tables and other temporary storage is also given. All tests have been performed using a Xeon E3-1240v3, single-threaded @ 3.8 GHz.

	Symbol	Value	Time	Memory
3j	$\begin{pmatrix} 15 & 30 & 40 \\ 2 & 2 & -4 \end{pmatrix}$	-0.01908157979919155	0.07 ms	58 kB
	$\begin{pmatrix} 200 & 200 & 200 \\ -10 & 60 & -50 \end{pmatrix}$	0.0007493927313989515	0.31 ms	730 kB
	$\begin{pmatrix} 50{,}000 & 50{,}000 & 50{,}000 \\ 1{,}000 & -6{,}000 & 5{,}000 \end{pmatrix}$	-1.116843916927519e-05	112 s	19 GB
6j	$\begin{Bmatrix} 8 & 8 & 8 \\ 8 & 8 & 8 \end{Bmatrix}$	-0.01265208072315355	0.06 ms	7.0 kB
	{ all 200 }	0.0001559032124132416	0.62 ms	1.0 MB
	{ all 600 }	-1.03981778344144e-07	6.1 ms	8.0 MB
	{ all 10,000 }	2.770313640470537e-08	8.2 s	1.5 GB
	{ all 50,000 }	3.997351841910046e-08	816 s	32 GB
9j	$\begin{Bmatrix} 8.5 & 9.5 & 7.0 \\ 12.5 & 8.0 & 8.5 \\ 8.0 & 10.5 & 9.5 \end{Bmatrix}$	0.0002812983019125448	0.08 ms	20 kB
	$\begin{Bmatrix} 100 & 80 & 50 \\ 50 & 100 & 70 \\ 60 & 50 & 100 \end{Bmatrix}$	1.055977980657612e-07	1.9 ms	0.50 MB
	{ all 200 }	1.278335300545066e-07	0.15 s	1.6 MB
	{ all 1,000 }	1.749851385596156e-09	29.8 s	30 MB
	{ all 2,000 }	2.755181565857189e-10	358 s	109 MB

Wigner 3j: `WIGXJPF` - fast and accurate

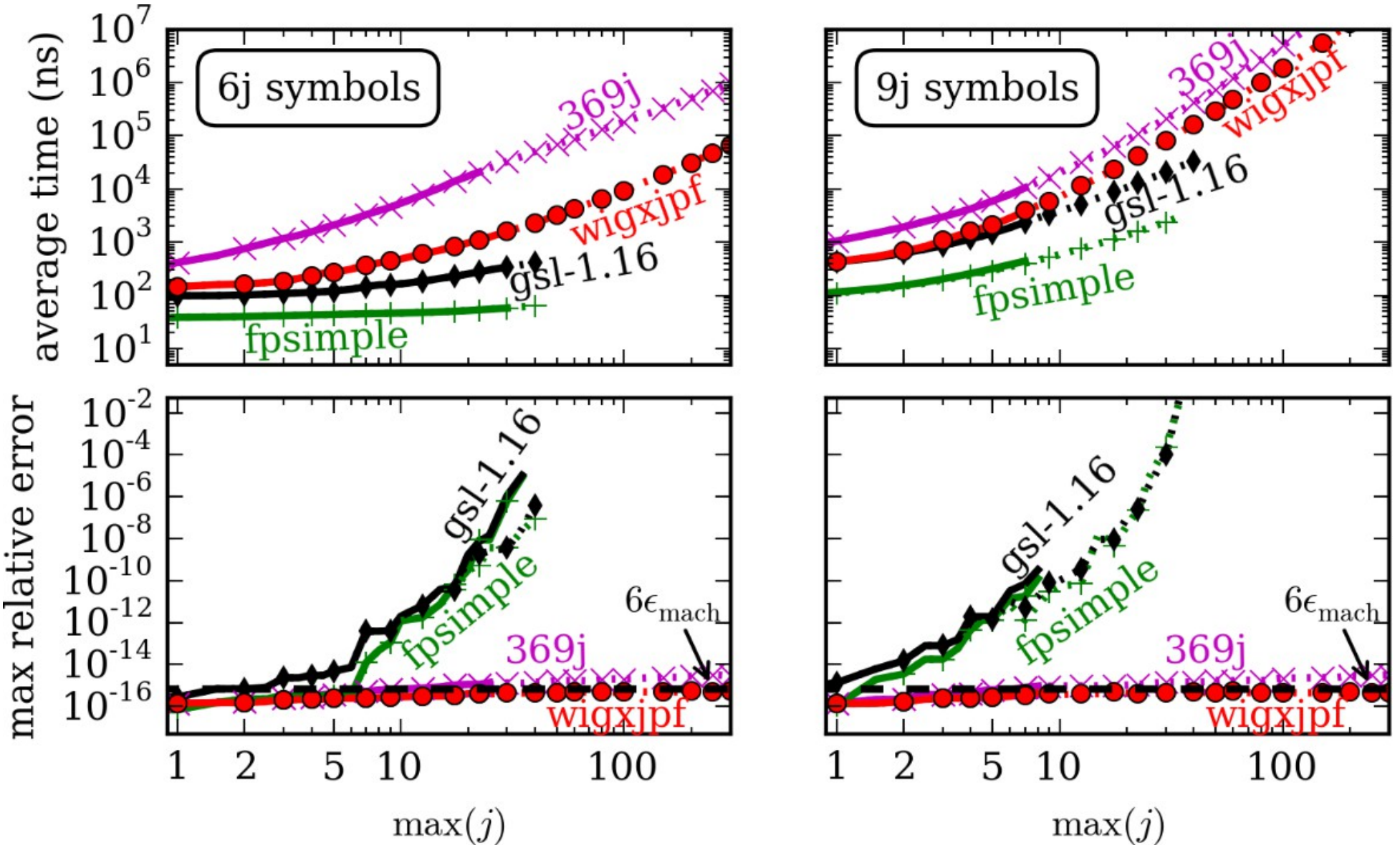


Fast (new) floating point implementation.
Not so accurate.

[L. Wei, Comput. Phys. Commun. 120, 222 (1999).]
Accurate.

GSL: GNU scientific library
http://www.gnu.org/software/gsl

Wigner 6j, 9j: `WIGXJPF` - fast and accurate



When fast is not quick enough

Store precalculated values in tables:

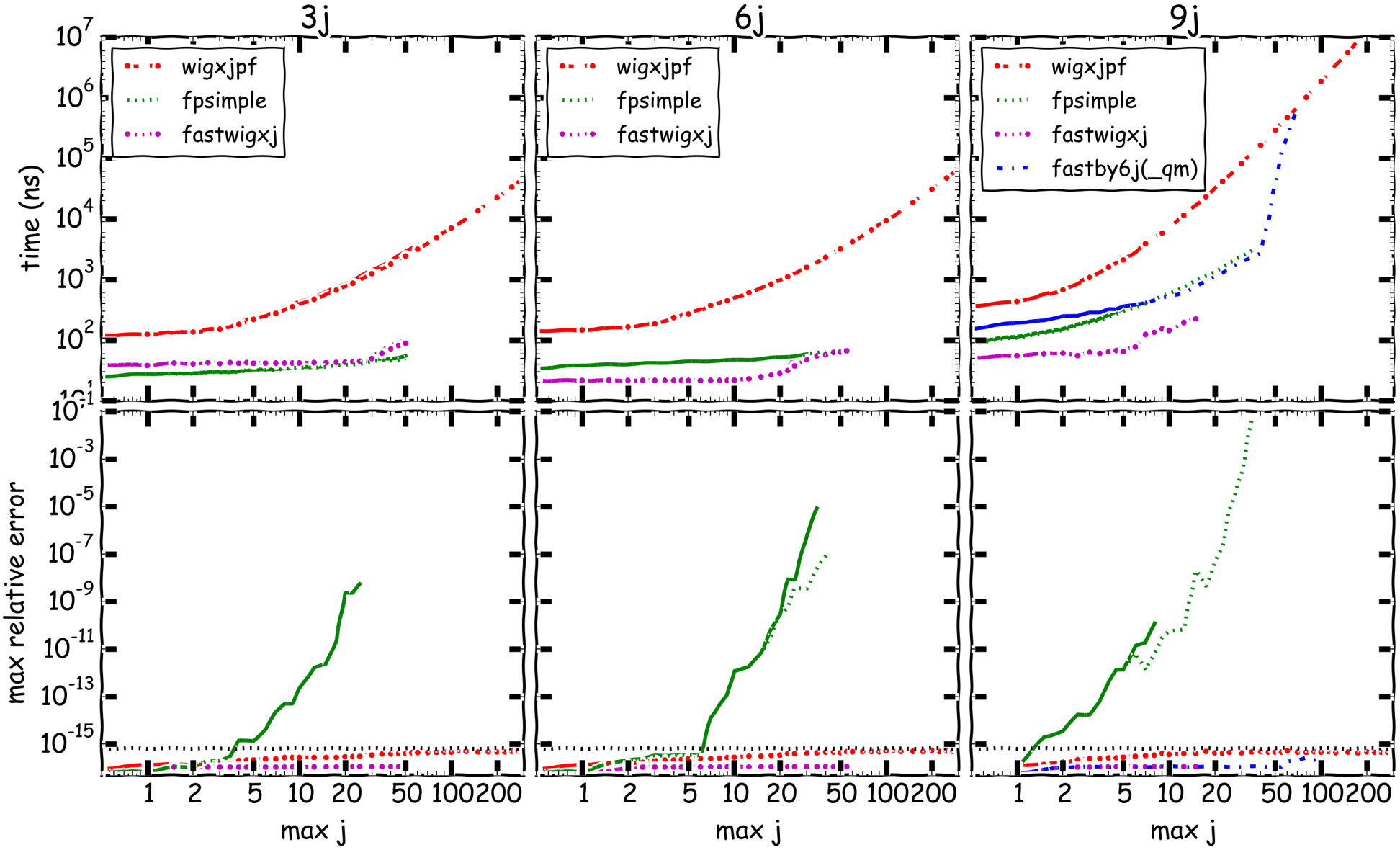
- Speed-up ~ 5x – 10x
- Full (floating-point) accuracy

	3j	6j	9j
1M	12	7½	4
10M	20	12	5½
1G	52½	28	9½
10G	-	40	13½

Use symmetries to reduce table sizes:

- 3j and 6j by clever formulas giving direct index:
  [J. Rasch and A. C. H. Yu, SIAM J. Sci. Comput., 25 (2003), pp. 1416-1428.]
  - limited by *random* memory lookup time **~90 ns** (large table)
- 9j by **explicit** generation of all permutations; hash table lookup
  - 72 permutations
  - performed by 300-500 machine instructions
  - in less than **50 ns** on modern hardware
  - subsequent *random* memory lookup **~110 ns** (large table)

FASTWIGXJ – precalculated WIGXJPF symbols



Factorials are FUN!

Thank you!

Libraries are available!
License: LGPLv3

`WIGXJPF:`
http://fy.chalmers.se/subatom/wigxjpf/
(Interfaces for C, FORTRAN, Python)

`FASTWIGXJ:`
http://fy.chalmers.se/subatom/fastwigxj/
(Interfaces for C, FORTRAN)

Publications:

eprint arXiv:1504.08329

Submitted to SIAM J. Sci. Comput.

In preparation.