Machine-independent tables

Tables without a cost column, written once for all machines; the cross-machine comparison of the accuracy fields is stated in README.md when two CPU architectures are present.

8 table(s) in this directory. Each is shown under the file that produces it, at the manuscript's column width, so a table that does not fit is visible here.

`tab-baselines.tex`

Table 1: Upstream release or commit of every external implementation of the benchmark, with the SHA-256 digest of each pinned upstream file (first 12 hexadecimal digits; in full in the `.csv` and in `references/UPSTREAM.tsv`), how it is obtained, what the runs used and what this project changes.

implementation	upstream release, commit	file, SHA-256	obtained	in the runs	changed by this project
bessels.jl	main branch, `009b615347d9`	src/ BesselFunctions/ besselk.jl `3560c95f769a`	make references (references/ fetch.sh), sha256 checked	C port of this commit	C port (references/ bessels_jl/port/ cpu_bessels_jl.c); libm tgamma for gamma
bessels.jl-old	v0.2.8, `1475e9e57464`	src/besselk.jl `e684931f3283`	make references (references/ fetch.sh), sha256 checked	C port of this commit	C port (references/ bessels_jl/port/ cpu_bessels_old.c) with the main branch's large-argument cut-off; libm tgamma for gamma
kaust-128, kaust-40, kaust-modified	`94ec9ca4bb9d`	src/logbesselk.cu `5a51e1a5c7c0`; include/logbesselk.h `650bafdad9c4`	make kaust DOWN-LOAD=yes after an explicit yes, sha256 checked, not redistributed	port generated from these files	C and CUDA port generated by references/ kaust_logbesselk/ derive_port.py, with the hooks of kaust-modified, whose changes references/kaust_ logbesselk/kaust_ port.c makes: adaptive integration interval, 128 bins, decimal exponent applied, log cosh without cosh
boost	1.75.0, `a0a97b768afd`	boost_1_75_0.tar.bz2 `953db31e016d`	a system installation, else make boost-headers (references/ fetch_boost.sh), sha256 checked	1.75.0, headers / usr/include (system), package boost-devel-1.75.0-11.el9_ 6.x86_64	none; an error policy that returns instead of throwing
boost-gpu, boost-gpu-modified	1.87.0, `f1b0b95a04f5`	boost_1_87_0.tar.bz2 `af57be25cb4c`	a system installation or environment module, else make boost-headers (references/ fetch_boost.sh), sha256 checked	1.87.0, headers / opt/bwhpc/ common/lib/ boost/1.87.0 (BOOST_ ROOT)	boost-gpu-modified: the recurrence in place of the Hankel shortcut (benchmark/ common/boost_ adapted.h)
gsl	2.6, `8f0165f5cb2a`	gsl-2.6.tar.gz `b782339fc7a3`	a system GSL found by pkg-config, else make externals (references/ fetch_gsl.sh), sha256 checked	2.6, package gsl-2.6-7.el9.x86_ 64	none; the abort-on-error handler disabled
std-cyl_bessel_k	GCC 14.2.0 (libstdc++ 14), `04696df09633`	libstdc++-v3/ include/tr1/ modified_bessel_ func.tcc	the standard library of the C++ compiler	libstdc++ 14 (20240801), gcc 14.2.0	none; a wrapper that returns instead of throwing
scipy (opt-in scipy stage)	1.18.1	scipy-1.18.1.tar.gz `52c4b7422442`	make scipy-headers (references/ fetch_scipy.sh; make pipeline SCIPY=yes), sha256 checked, xsf headers only, not redistributed	1.18.1, headers placed from the verified archive (references/ scipy/upstream)	none; compiled with -ffp-contract=off (kv and kve equal to the wheel's in make scipy-check)

–: not in the runs read. kaust-modified and boost-gpu-modified are recorded as `kaust-adaptive` and

`tab-datasets.tex`

Table 2: Benchmark datasets and their sizes, in points, for each argument convention.

dataset	(z, ν) supplied	(h, ν, s) — z computed
Grid	262 144	262 144
Random	250 000	250 000
Small z	100 000	100 000
Near-int ν	100 200	100 200
Half-int ν	100 000	100 000
Large z	100 000	100 000
Large ν	150 000	150 000
Seams	100 000	100 000
Size / rand.	1 048 576	1 048 576
Size / half	1 048 576	1 048 576

Size: the datasets of the size-scaling benchmarks.

tab-environment.tex

Table 3: Benchmark environments used for the measurements, one column per machine class. Common to all machines: 2 sockets, Linux 5.14.0-570.149.1.el9_6 kernel, scalar_portable and matched backends built without FMA, SIMD extensions SSE2, AVX, AVX2, FMA, AVX-512 (F, DQ, VL).

Field	Intel	Intel+GPU	AMD
Machine	xeon-platinum-8358 (all, fold, levin, threads)	xeon-platinum-8358 (gpu)	epyc-9454 (all, fold, levin, likelihood, scipy, threads)
CPU	Intel Xeon Platinum 8358 @ 2.60GHz	Intel Xeon Platinum 8358 @ 2.60GHz	AMD EPYC 9454
Physical cores	64	64	96
Hardware threads	128	128	192
NUMA domains	4	4	8
OS	RHEL 9.6	RHEL 9.6	RHEL 9.6
Compiler, library	GCC 11.5.0	GCC 11.5.0	GCC 11.5.0
Compiler, C++ suppliers	GCC 14.2.0	–	GCC 14.2.0
LAPACK, likelihood stage	–	–	OpenBLAS 0.3.26, Cooperlake, one thread
GPU	None	NVIDIA A100 80GB PCIe	None
Compute capability	–	8.0	–
Device memory	–	81920 MiB	–
Driver	–	615.71.09	–
CUDA driver API	–	13.4	–
nvcc	not built	12.8.61	not built
nvcc host compiler	–	GCC 14.2.0	–
Built for	–	library sm_80 (ptx sm_80); benchmark_gpu sm_80 (ptx sm_80); suppliers_gpu sm_80 (ptx sm_80)	–
ECC	–	Enabled	–
Persistence mode	–	Enabled	–
Max SM clock	–	1410 MHz	–
Max memory clock	–	1512 MHz	–
Power limit	–	300.00 W	–
Throttle before run	–	GpuIdle	–
Boost, CPU row	1.75.0	–	1.75.0
Boost, GPU row	–	1.87.0	–
GSL	2.6	–	2.6
L3 cache (total)	96 MiB (2 instances)	96 MiB (2 instances)	512 MiB (16 instances)
cpufreq driver	intel_pstate	intel_pstate	amd-pstate
cpufreq governor	performance	performance	performance
cpufreq range	0.8-3.4 GHz	0.8-3.4 GHz	0.4-3.812 GHz
Clock in the timed rows	3400 MHz median (799..3400 MHz)‡	3300 MHz median (3299..3399 MHz)	3800 MHz median (3799..3800 MHz)‡
Turbo boost	On	On	On
L1d cache	48 KiB per core	48 KiB per core	32 KiB per core
L1i cache	32 KiB per core	32 KiB per core	32 KiB per core
L2 cache	1.25 MiB per core	1.25 MiB per core	1 MiB per core
L3 cache	48 MiB per 32 cores	48 MiB per 32 cores	32 MiB per 6 cores

Machine: the machine and the stages run on it. ‡: not the same in every run on that machine class; the value of its run with the most stages is shown. –: not applicable or not recorded.

`tab-exclusions-h_nu_s.tex`

Table 4: Points at which an implementation returned no finite number, for the (h, ν, s) datasets. Results were obtained on the AMD CPU.

implementation	Grid	Random	Small z	Near-int ν	Half-int ν	Large z	Large ν	Seams	points
libmatern (avx512)	0	0	0	0	0	0	0	0	varies
bessels.jl	0	0	11 913	0	0	0	48 712	0	varies
bessels.jl-old	0	0	11 973	0	0	0	48 712	0	varies
boost	0	0	11 856	0	0	0	47 474	0	varies
gsl	0	0	0	0	0	0	0	0	varies
kaust-128	0	0	0	0	0	21 076	86 860	0	varies
kaust-40	0	0	0	0	0	21 076	86 860	0	varies
kaust-modified	0	0	0	0	0	0	0	0	varies
std-cyl_bessel_k	0	0	11 856	0	0	0	47 474	0	varies

points: size of the datasets; *varies*: the datasets differ in size. `kaust-modified`: an adjusted configuration of the KAUST code, more accurate than the code as released (the adaptive integration range of its paper in place of $t \in [0, 9]$, 128 panels, the decade factor of the small-x recurrence applied, $\log\cosh$ without the intermediate cosh).

`tab-exclusions-z_nu.tex`

Table 5: Points at which an implementation returned no finite number, for the (z, ν) datasets. Results were obtained on the AMD CPU.

implementation	Grid	Random	Small z	Near-int ν	Half-int ν	Large z	Large ν	Seams	points
libmatern (avx512)	0	0	0	0	0	0	0	0	varies
bessels.jl	0	0	11 913	0	0	0	48 712	0	varies
bessels.jl-old	0	0	11 973	0	0	0	48 712	0	varies
boost	0	0	11 856	0	0	0	47 474	0	varies
gsl	0	0	0	0	0	0	0	0	varies
kaust-128	0	0	0	0	0	21 076	86 860	0	varies
kaust-40	0	0	0	0	0	21 076	86 860	0	varies
kaust-modified	0	0	0	0	0	0	0	0	varies
std-cyl_bessel_k	0	0	11 856	0	0	0	47 474	0	varies

points: size of the datasets; *varies*: the datasets differ in size. `kaust-modified`: an adjusted configuration of the KAUST code, more accurate than the code as released (the adaptive integration range of its paper in place of $t \in [0, 9]$, 128 panels, the decade factor of the small-x recurrence applied, $\log\cosh$ without the intermediate cosh).

tab-medmax-h_nu_s.tex

Table 6: Median and maximum ULP error (median / maximum) over all scored points for the (h, ν, s) datasets, by implementation. Each external implementation is shown in its best assembly for each dataset, the one with the lowest maximum ULP error among those with the fewest non-finite results, direct on an exact tie. Results were obtained on the AMD CPU.

implementation	Grid	Random	Small z	Near-int ν	Half-int ν	Large z	Large ν	Seams	finite
libmatern (avx512)	1.17 / 19.9	1.12 / 36.3	0.483 / 6.70	0.879 / 24.8	0.609 / 10.9	0.723 / 20.0	0.734 / 43.6	1.17 / 34.5	8/8
bessels.jl	26.1 / 6804	21.4 / 620	1.00 / 1740†	5.74 / 3.77×10^{4}	43.4 / 1662	173 / 2306	179 / $1.82 \times 10^{4\dagger}$	4.62 / 1.26×10^{4}	6/8
bessels.jl-old	17.9 / 7209	23.3 / 1.51×10^{4}	57.0 / 1740†	24.4 / *Cap*	47.0 / 1713	192 / 3087	188 / $1.82 \times 10^{4\dagger}$	5.27 / 9218	5/8
boost	12.7 / 298	2.11 / 81.3	0.948 / 6.72†	1.03 / 49.4	1.54 / *Cap*	78.8 / *Cap*	162 / *Cap*†	1.39 / 1353	4/8
gsl	12.8 / 297	3.34 / 114	64.0 / 2283	2.58 / 749	40.6 / 1664	172 / 2306	397 / *Cap*	14.1 / 1724	7/8
kaust-128	33.8 / 3711	8.25 / *Cap*	5.94 / *Cap*	4.76 / *Cap*	8.03 / *Cap*	2.75×10^{11} / *Cap*†	59.6 / *Cap*†	6.50 / *Cap*	1/8
kaust-40	2.15×10^{13} / *Cap*	4.26×10^{10} / *Cap*	6.35 / *Cap*	1.05×10^{4} / *Cap*	2.06×10^{8} / *Cap*	2.35×10^{15} / *Cap*†	1.04×10^{13} / *Cap*†	4.63×10^{8} / *Cap*	0/8
kaust-modified	21.3 / 381	9.06 / 277	64.0 / 2090	4.73 / 325	45.9 / 1901	187 / 2942	345 / 2.96×10^{4}	5.86 / 1997	8/8
std-cyl_bessel_k	12.8 / 3800	4.54 / 2649	4.29 / 131†	5.51 / *Cap*	6.45 / *Cap*	79.2 / *Cap*	162 / *Cap*†	4.16 / 2908	3/8
Lowest external	boost/ gsl	boost	boost/ kaust-mod.	boost	boost/ bessels.jl	boost/ bessels.jl	kaust-128/ kaust-mod.	boost	–
Advantage factor	lib 10.8/14.9	lib 1.89/2.24	lib 1.96/312	lib 1.18/2.00	lib 2.53/153	lib 109/115	lib 81.2/679	lib 1.19/39.1	–

Cap: a maximum at or above the 10^{6} ULP cap; the value is in the `.csv`. †: maximum over a reduced population, non-finite results excluded. *finite*: number of datasets with a finite result at every point and a maximum below 10^{6} ULP. *Lowest external*: the external implementation with the lowest median and the one with the lowest maximum (median/maximum; one name where both coincide). The median is ranked over all external implementations, rows marked † or *Cap* included; the maximum only over those with a finite result at every point and a maximum below the 10^{6} ULP cap. *Advantage factor*: for the median and the maximum, the ratio of the larger to the smaller of the lowest external value and this library's, labelled with the implementation it favours (`lib`: this library). `kaust-modified`: an adjusted configuration of the KAUST code, more accurate than the code as released (the adaptive integration range of its paper in place of $t \in [0, 9]$, 128 panels, the decade factor of the small-x recurrence applied, $\log\cosh$ without the intermediate cosh).

tab-medmax-z_nu.tex

Table 7: Median and maximum ULP error (median / maximum) over all scored points for the (z, ν) datasets, by implementation. Each external implementation is shown in its best assembly for each dataset, the one with the lowest maximum ULP error among those with the fewest non-finite results, direct on an exact tie. Results were obtained on the AMD CPU.

implementation	Grid	Random	Small z	Near-int ν	Half-int ν	Large z	Large ν	Seams	finite
libmatern (avx512)	1.10 / 12.5	1.02 / 21.8	0.483 / 10.4	0.864 / 17.7	0.600 / 10.4	0.707 / 14.7	0.729 / 25.1	1.06 / 22.4	8/8
bessels.jl	22.0 / 6804	2.96 / 886	1.00 / 1740†	5.09 / 3.77×10^4	40.9 / 1413	125 / 1542	177 / 1.81×10^{4}†	2.49 / 1.26×10^4	6/8
bessels.jl-old	2.31 / 7209	22.4 / 1.51×10^4	57.0 / 1740†	24.2 / *Cap*	44.6 / 1539	165 / 3020	186 / 1.81×10^{4}†	3.21 / 9218	5/8
boost	0.740 / 6.18	0.773 / 6.43	0.947 / 6.72†	0.893 / 6.22	1.01 / *Cap*	0.905 / *Cap*	160 / *Cap*†	0.770 / 22.6	4/8
gsl	1.34 / 25.8	1.80 / 114	64.0 / 2283	2.35 / 749	38.3 / 1248	124 / 1447	392 / *Cap*	13.5 / 1060	7/8
kaust-128	30.5 / 3754	7.45 / *Cap*	5.94 / *Cap*	4.69 / *Cap*	7.71 / *Cap*	2.75×10^{11} / *Cap*†	58.4 / *Cap*†	6.15 / *Cap*	1/8
kaust-40	2.15×10^{13} / *Cap*	4.26×10^{10} / *Cap*	6.34 / *Cap*	1.05×10^4 / *Cap*	2.06×10^{8} / *Cap*	2.35×10^{15} / *Cap*†	1.04×10^{13} / *Cap*†	4.63×10^{8} / *Cap*	0/8
kaust-modified	17.1 / 315	8.53 / 277	64.0 / 2090	4.71 / 325	44.3 / 1398	167 / 2045	342 / 2.59×10^4	5.61 / 974	8/8
std-cyl_bessel_k	1.39 / 3799	21.1 / 2647	4.28 / 131†	5.08 / *Cap*	3.17 / *Cap*	2.13 / *Cap*	160 / *Cap*†	2.53 / 2906	3/8
Lowest external	boost	boost	boost/ kaust-mod.	boost	boost/ gsl	boost/ gsl	kaust-128/ kaust-mod.	boost	–
Advantage factor	boost 1.49/2.02	boost 1.32/3.40	lib 1.96/201	lib 1.03/ boost 2.84	lib 1.68/120	lib 1.28/98.7	lib 80.1/1032	boost 1.37/ lib 1.01	–

Cap: a maximum at or above the 10^6 ULP cap; the value is in the `.csv`. †: maximum over a reduced population, non-finite results excluded. *finite*: number of datasets with a finite result at every point and a maximum below 10^6 ULP. *Lowest external*: the external implementation with the lowest median and the one with the lowest maximum (median/maximum; one name where both coincide). The median is ranked over all external implementations, rows marked † or *Cap* included; the maximum only over those with a finite result at every point and a maximum below the 10^6 ULP cap. *Advantage factor*: for the median and the maximum, the ratio of the larger to the smaller of the lowest external value and this library's, labelled with the implementation it favours (`lib`: this library). `kaust-modified`: an adjusted configuration of the KAUST code, more accurate than the code as released (the adaptive integration range of its paper in place of $t \in [0, 9]$, 128 panels, the decade factor of the small-x recurrence applied, $\log\cosh$ without the intermediate cosh).

`tab-references.tex`

Table 8: Count of zero and subnormal reference values for each dataset, identical for both argument conventions.

dataset	points	$M = 0$	subnormal M
Grid	262 144	0	0
Random	250 000	0	0
Small z	100 000	0	0
Near-int ν	100 200	0	0
Half-int ν	100 000	0	0
Large z	100 000	432 (0.4 %)	901 (0.9 %)
Large ν	150 000	6 483 (4.3 %)	257 (0.2 %)
Seams	100 000	0	5 875 (5.9 %)

points: size of the dataset. $M = 0$: the stored reference is 0, the true value lies below 2^{-1075}. Subnormal: $0 < |M| < 2^{-1022}$. Percentages are shares of the points.