Accurate Computation of the Logarithm of Modified Bessel Functions on GPUs

Andreas L. Plesner[1], Hans Henrik Brandenborg Sørensen[2], Søren Hauberg[2]

[1]ETH Zurich, **aplesner@ethz.ch**

[2]Technical University of Denmark, {hhbs, sohau}@dtu.dk

We want to compute log $I_v(x)$

For x = 0.5 and v = 129.25

- SciPy: -681.6595192185806
- MATLAB: -681.6595192185806
- boost: -681.6595192185805
- GNU Scientific Library (GSL): -681.6595192185805
- Mathematica: -681.6595192185806
- Wolfram|Alpha: -681.6595192185805

We want to compute log $I_v(x)$

For x = 0.5 and v = 139.25

- SciPy: -Infinity
- MATLAB: -Infinity
- boost: -744.4400719213812
- GNU Scientific Library (GSL): -Infinity
- Mathematica: -744.4400719213812
- Wolfram|Alpha: -744.5544364434038

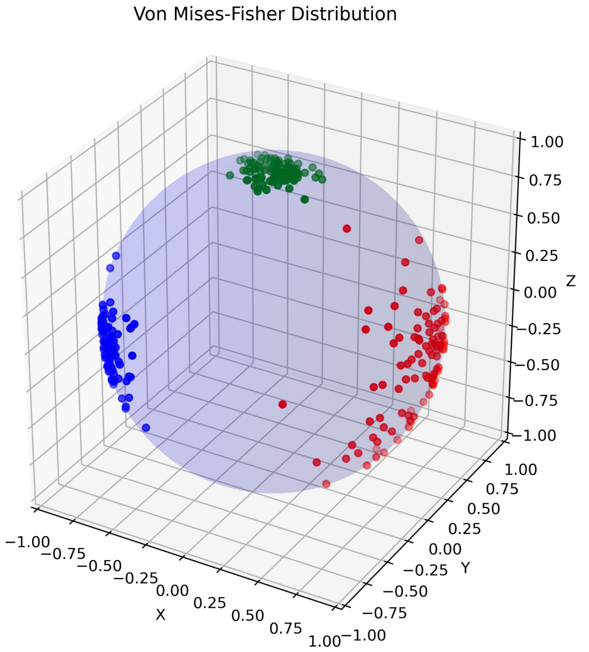
A brief introduction to (modified) Bessel functions

Bessel's Modified Differential Equation

$$x^2 \frac{\mathrm{d}^2 y}{\mathrm{d}x^2} + \frac{\mathrm{d}y}{\mathrm{d}x} - \left(x^2 + v^2\right) y = 0$$

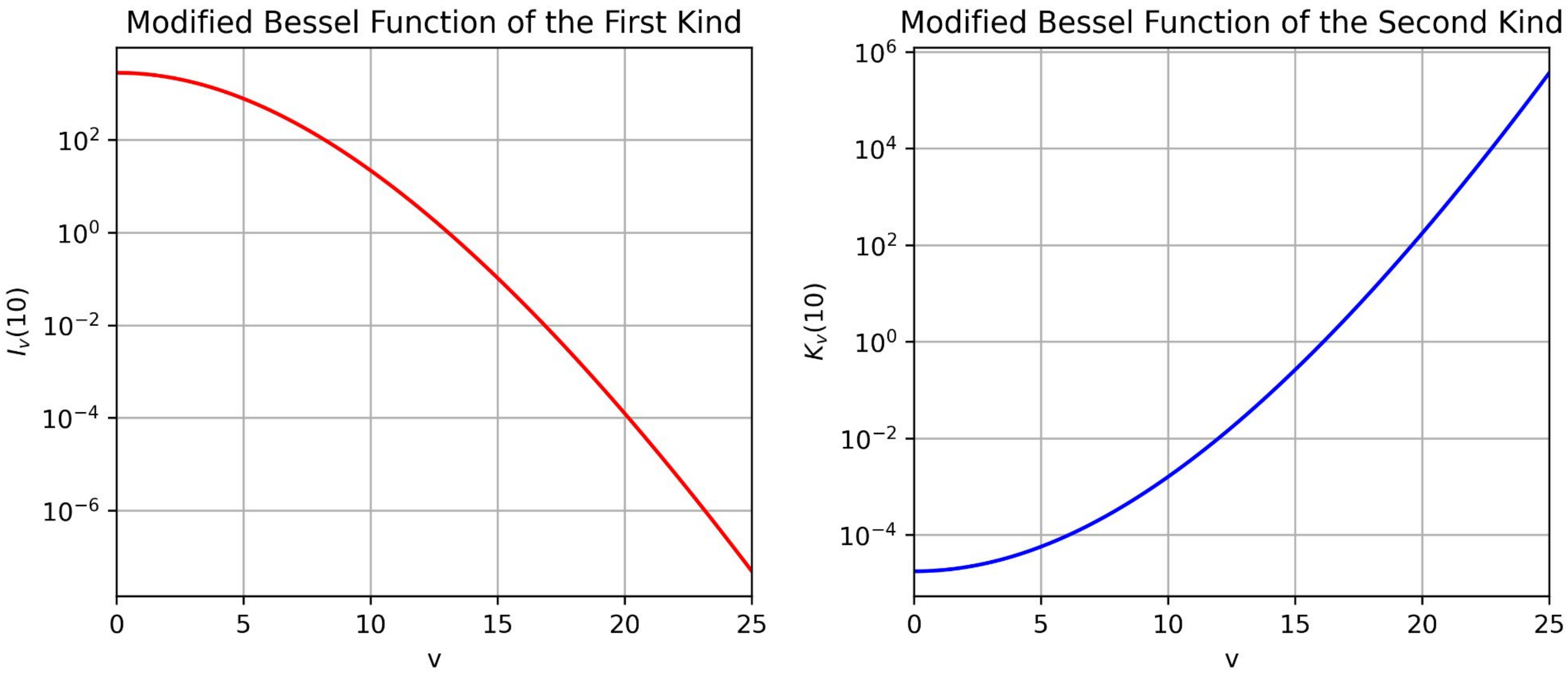
$$I_v(x) = \left(\frac{1}{2} z\right)^v \sum_{k=0}^{\infty} \frac{\left(\frac{z}{4}\right)^k}{k!\Gamma(v+k+1)}$$

$$K_v(x)$$



$$f(x; \mu, \kappa) = \frac{\kappa^{p/2-1}}{(2\pi)^{p/2} I_{p/2-1}(\kappa)} \exp\left(\kappa \mu^\top x\right)$$

A brief introduction to (modified) Bessel functions

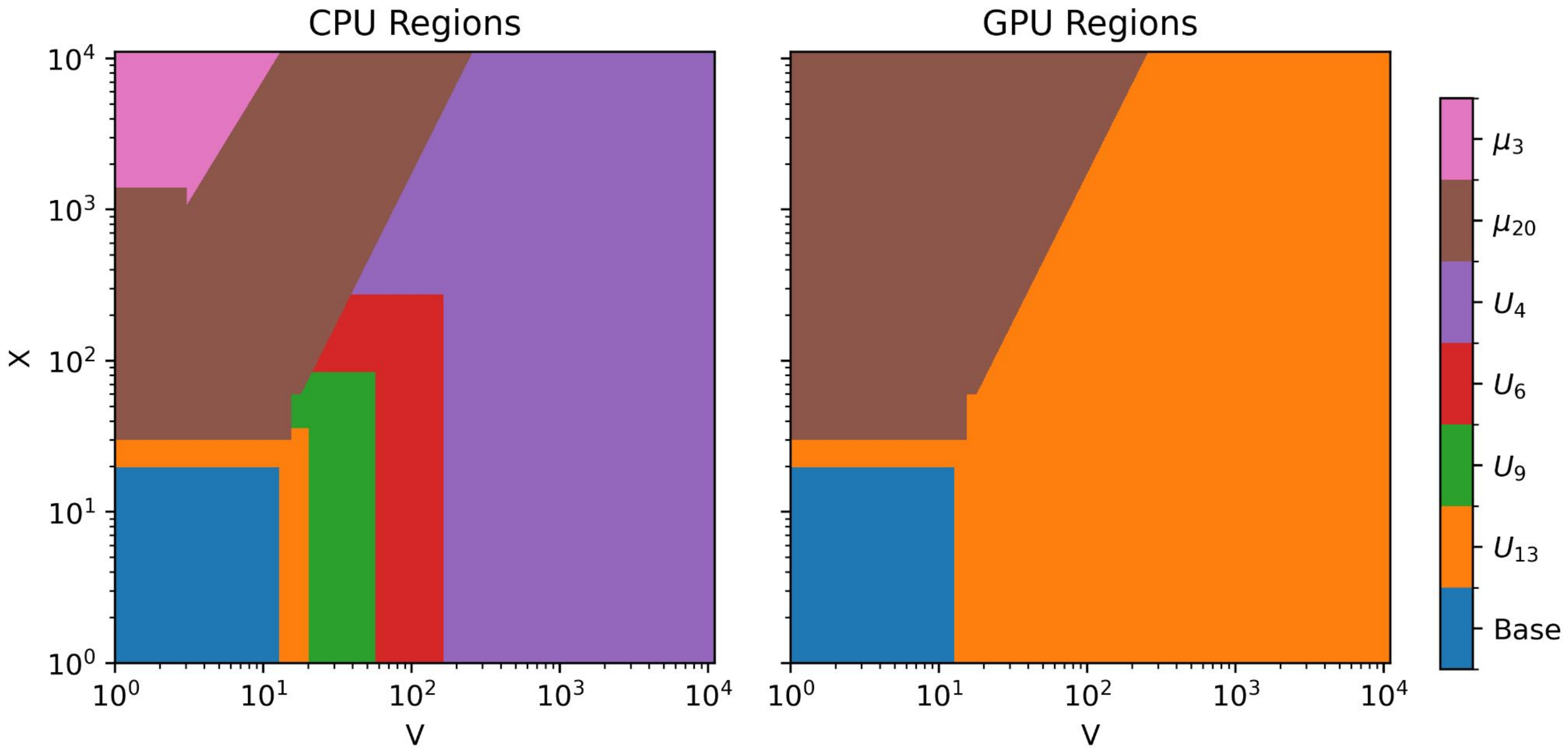


What do these libraries do and how can we do better?

- Scaling the Bessel functions with exponentials
- Do intermediate computations in log scale
- Use asymptotic expressions
- Sort input values to prevent thread divergence on GPUs

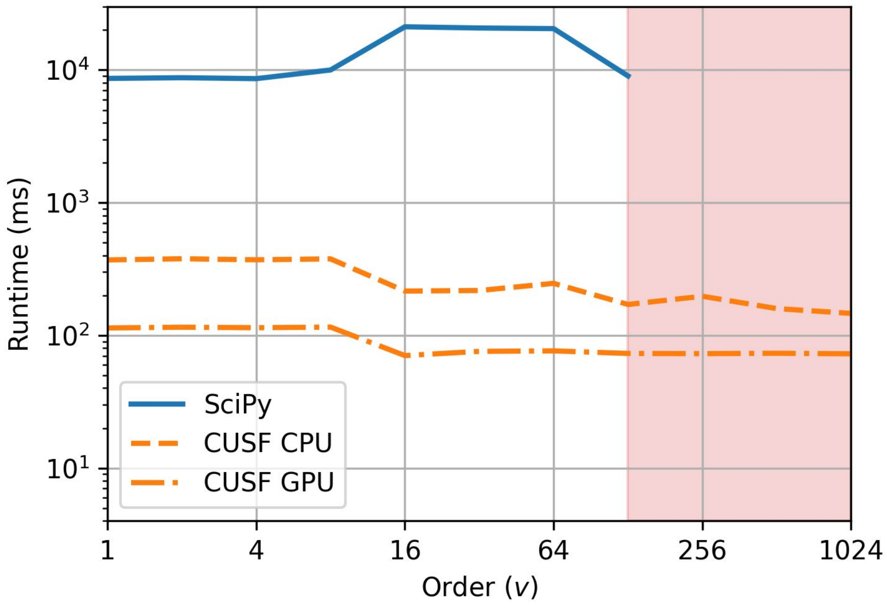
Regions of the Different Expressions

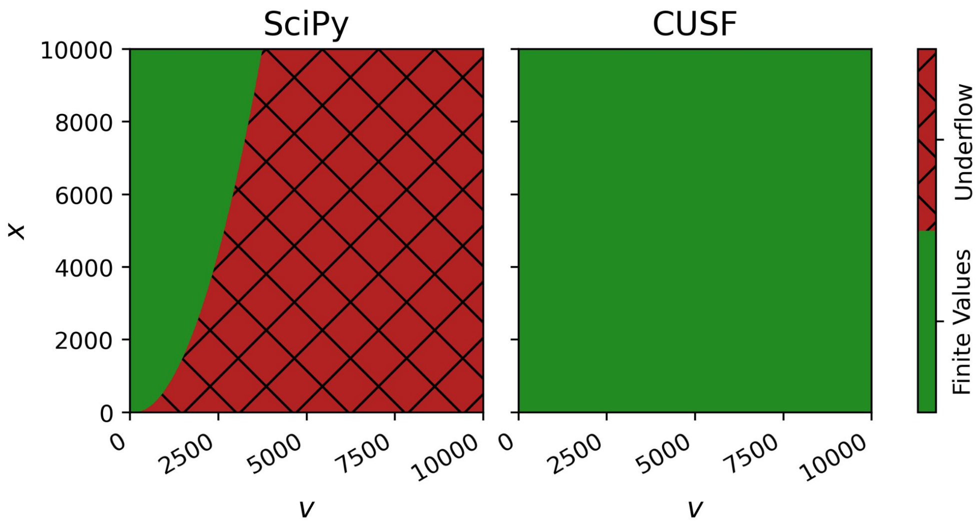
$$I_v(x) \approx \frac{\exp(x)}{\sqrt{2\pi x}}\left(1+\sum_{k=1}^{\infty}(-1)^k\frac{\prod_{j=1}^{k}\left(\mu-(2j-1)^2\right)}{k!(8x)^k}\right), \mu = 4v^2$$



How do we compare against SciPy for log $I_v(x)$

Median speedups of 35x and 77x for CPU and GPU, resp.
Maximum speedups of 98x and 300x for CPU and GPU, resp.



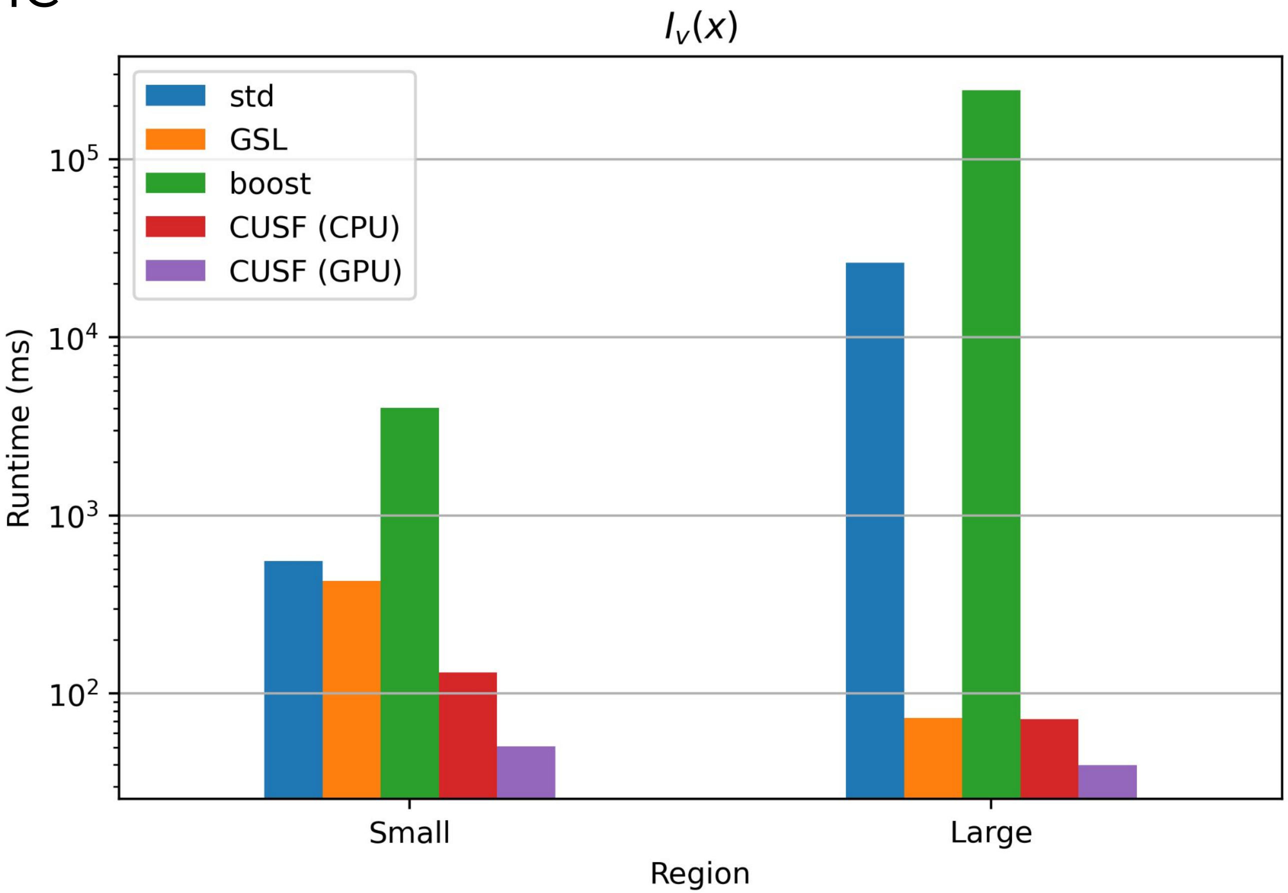


Accuracy and robustness

Function	Region	Metric	Library				
			std	GSL	Boost	CUSF (CPU)	CUSF (GPU)
$\log I_v(x)$	Small	Robustness	100%	99.98%	**100%**	100%	100%
		Median	4.04×10^{-16}	1.34×10^{-16}	0.0	2.12×10^{-16}	2.08×10^{-16}
		Maximum	2.77×10^{-6}	2.03×10^{-7}	4.10×10^{-8}	8.30×10^{-4}	8.30×10^{-4}
	Large	Robustness	0.50%	44.13%	1.98%	**100%**	**100%**
		Median	1.20×10^{-16}	0.00	0.00	2.40×10^{-16}	2.28×10^{-16}
		Maximum	1.20×10^{-5}	1.46×10^{-15}	0.00	2.98×10^{-13}	2.98×10^{-13}

Runtime

Median speedups of 17x and 45x for CPU and GPU, resp.
Maximum speedups of 3403x and 6150x for CPU and GPU, resp.



Using CUSF with PyTorch

Imports

```
import numpy as np
import torch
from scipy.special import ive as scipy_ive
from cusf.bessel import iv_log as cusf_iv_log
```

Helper function

```
def scipy_iv_log(nu, z):
    np_result = np.log(scipy_ive(nu.cpu().numpy(), z.cpu().numpy())) + z.cpu().numpy()
    return torch.from_numpy(np.asarray(np_result)).to(device=z.device)
```

Data loading

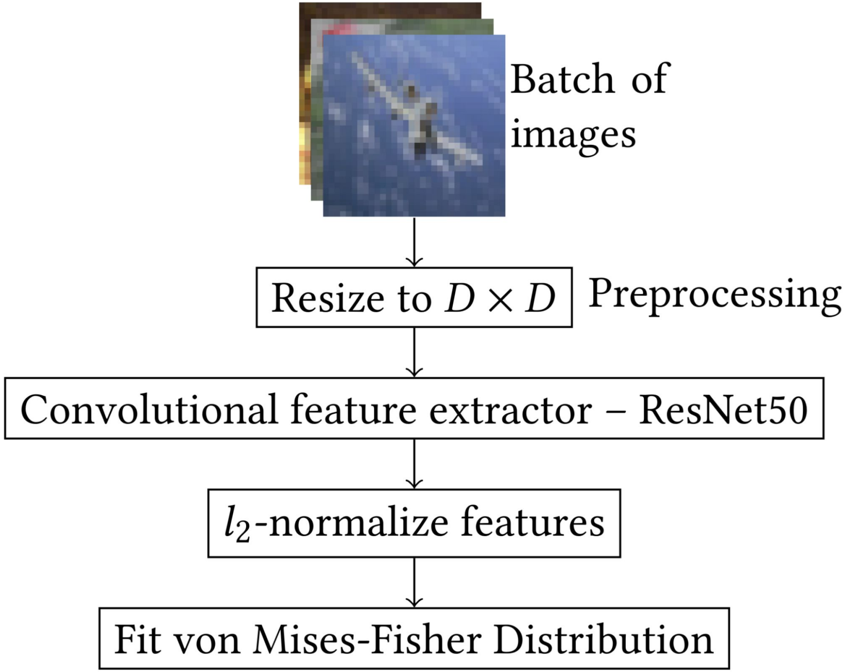
```
# Generate some data
torch.manual_seed(0)
device = torch.device(type='cuda')
N = 2000
M = 2000
z = torch.rand(N, M, device=device) * 150
nu = torch.rand_like(z) * 150
```

Doing stuff

```
# Evaluate
scipy_res = scipy_iv_log(nu=nu, z=z)
cusf_res = cusf_iv_log(nu=nu, z=z)
```

Estimating von Mises-Fisher

Pipeline



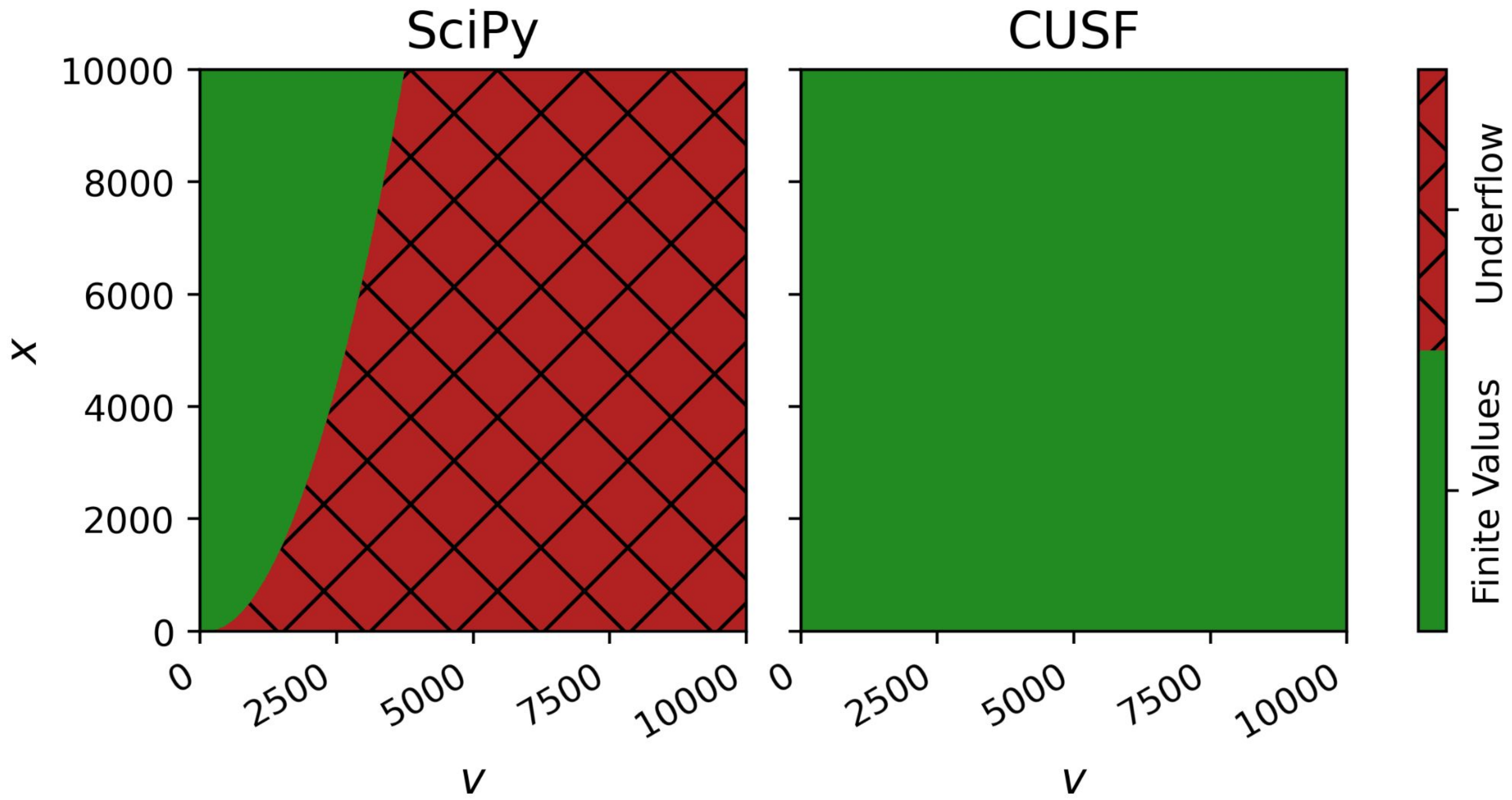
Results

# of features	2048	8192	32768
Gradient free estimate	298.9091	1577.135	6681.31
Gradient estimate	298.9098	1577.405	6668.07
$\widehat{\kappa_0}$	298.9127	1577.412	6668.08
$\widehat{\kappa_1}$	298.9098	1577.405	6668.07
$\widehat{\kappa_2}$	298.9098	1577.405	6668.07

Is Mathematica the right benchmark?

Function	Region	Metric	Library				
			std	GSL	Boost	CUSF (CPU)	CUSF (GPU)
$\log I_v(x)$	Selected values	Robustness	100%	0%	100%	**100%**	**100%**
		Median	$2.28 \cdot 10^{-5}$	N/A	$2.28 \cdot 10^{-5}$	$1.53 \cdot 10^{-16}$	$1.53 \cdot 10^{-16}$
		Maximum	$8.30 \cdot 10^{-4}$	N/A	$8.30 \cdot 10^{-4}$	$3.07 \cdot 10^{-16}$	$3.07 \cdot 10^{-16}$

Conclusion



SciPy
CUSF
10000
8000
6000
4000
2000
0
x
0
2500
5000
7500
10000
v
0
2500
5000
7500
10000
v
Underflow
Finite Values

Questions?

Andreas L. Plesner

aplesner@ethz.ch

Comparison to CUDA

Precision

Function	Region	Metric	Library					
			std	GSL	Boost	CUDA	CUSF (CPU)	CUSF (GPU)
$\log I_0(x)$	Small	Robustness	**100%**	100%	100%	100%	100%	100%
		Median	0.00	0.00	0.00	0.00	0.00	0.00
		Maximum	$1.11 \cdot 10^{-14}$	$3.08 \cdot 10^{-9}$	$9.07 \cdot 10^{-10}$	$9.07 \cdot 10^{-10}$	$3.68 \cdot 10^{-13}$	$3.68 \cdot 10^{-13}$
	Large	Robustness	61%	**100%**	61%	61%	100%	100%
		Median	0.00	0.00	0.00	0.00	0.00	0.00
		Maximum	$1.71 \cdot 10^{-16}$	$2.16 \cdot 10^{-16}$	$2.15 \cdot 10^{-16}$	$2.15 \cdot 10^{-16}$	$2.22 \cdot 10^{-16}$	$2.22 \cdot 10^{-16}$

Speed

Function	Region	Library					
		std	GSL	Boost	CUDA	CUSF (CPU)	CUSF (GPU)
$\log I_0(x)$	Small	3512.69 ± 6.57	915.87 ± 10.33	903.93 ± 26.38	$\mathbf{61.56 \pm 1.88}$	1665.29 ± 5.96	433.00 ± 18.63
	Large	19560.23 ± 14.13	929.41 ± 24.00	2080.96 ± 33.07	$\mathbf{50.15 \pm 0.26}$	301.89 ± 25.13	187.14 ± 21.57