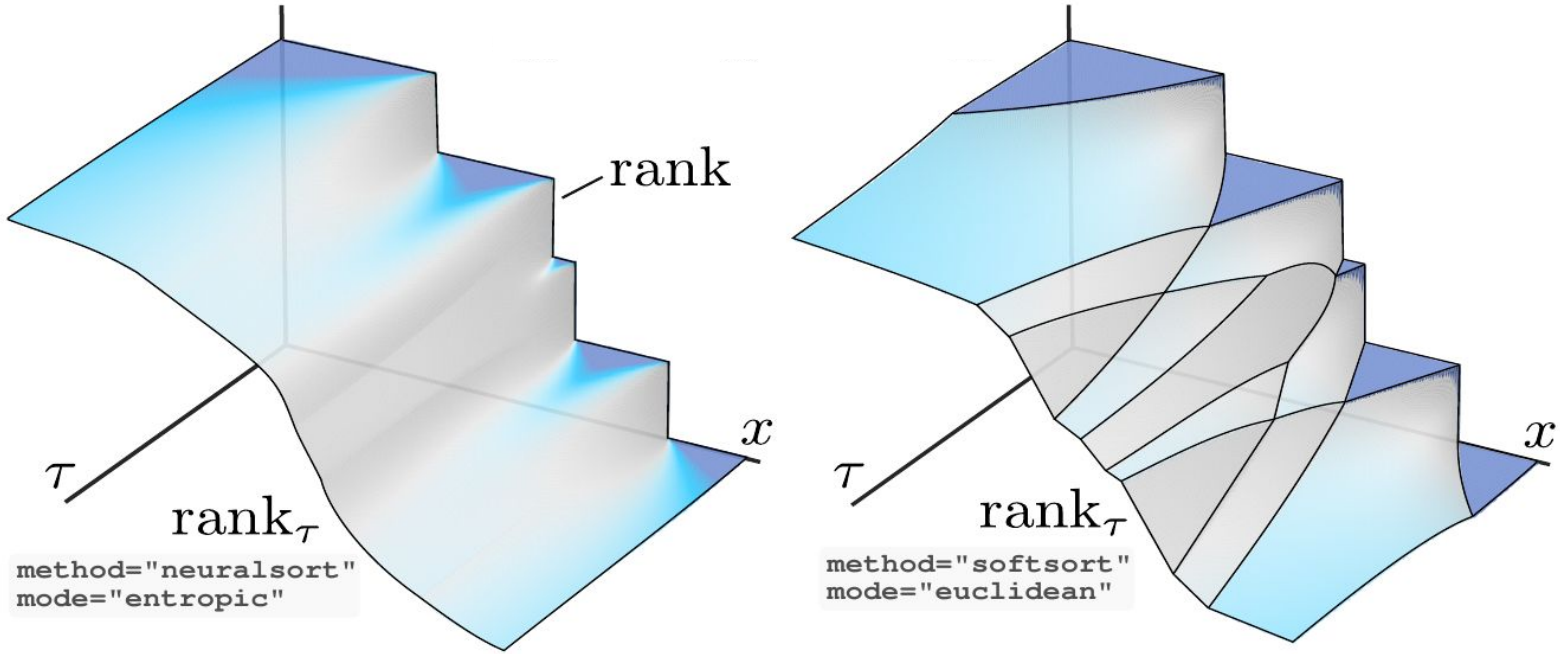


SoftJAX & SoftTorch

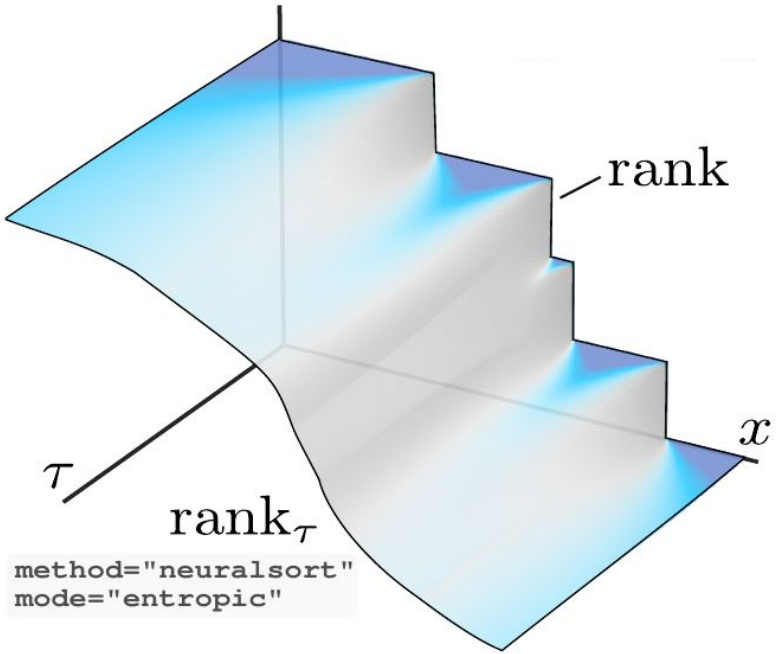
Empowering Autodiff with Informative Gradients

Dr. A. René Geist

Autonomous Learning

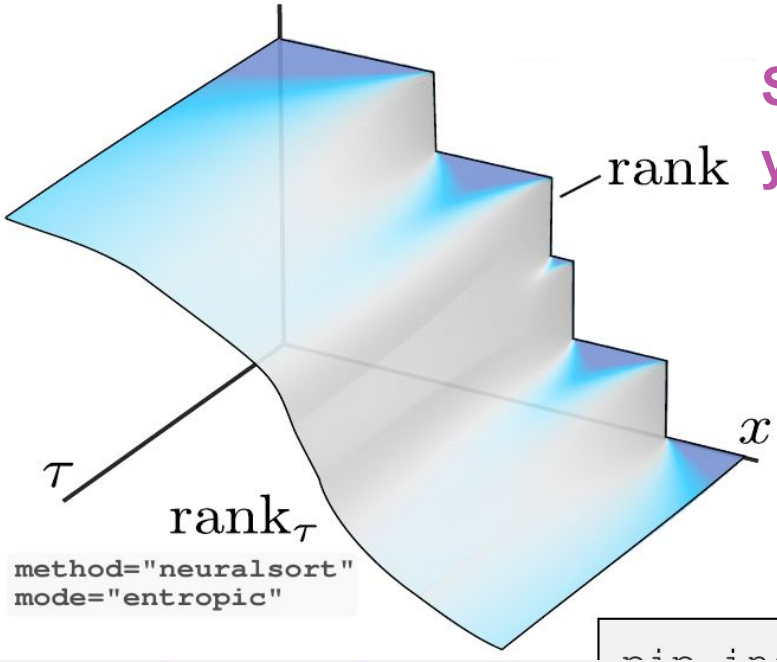
EBERHARD KARLS UNIVERSITÄT TÜBINGEN

SoftJAX & SoftTorch, A Paulus, **AR Geist**, V Musil, S Hoffmann, O Beker, G Martius, 2026.




```
arr = jnp.array([x, -1.0, 0.5, 0.8, 2.0])
```

SoftJAX & SoftTorch, A Paulus, **AR Geist**, V Musil, S Hoffmann, O Beker, G Martius, 2026.



Some operations in JAX & PyTorch yield uninformative gradients.

Existing “soft operators” are spread out across repos.

```
pip install softjax
```

```
arr = jnp.array([x, -1.0, 0.5, 0.8, 2.0])
```

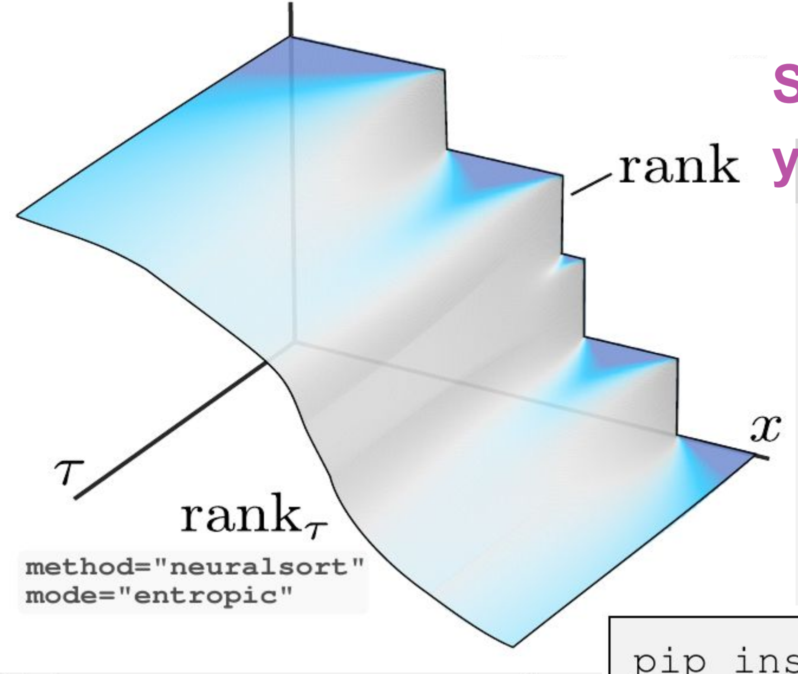
```
import softjax as sj
arr = jnp.array([x, -1.0, 0.5, 0.8, 2.0])
ranks = sj.rank(arr, method="neuralsort",
                mode="smooth")[0]
```

SoftJAX & SoftTorch, A Paulus, **AR Geist**, V Musil, S Hoffmann, O Beker, G Martius, 2026.

so… …gates for…

Some operations in JAX & PyTorch yield uninformative gradients.

Existing “soft operators” are spread out across repos.



Axiswis…

argmi…
min
argso…
sort
argqu…
quant…
argme…
media…
argto…
top_k
argrank
rank

not_…
iscl…

Logical:

logical_and
logical_or
logical_xor
logical_not
any
all

Selection:

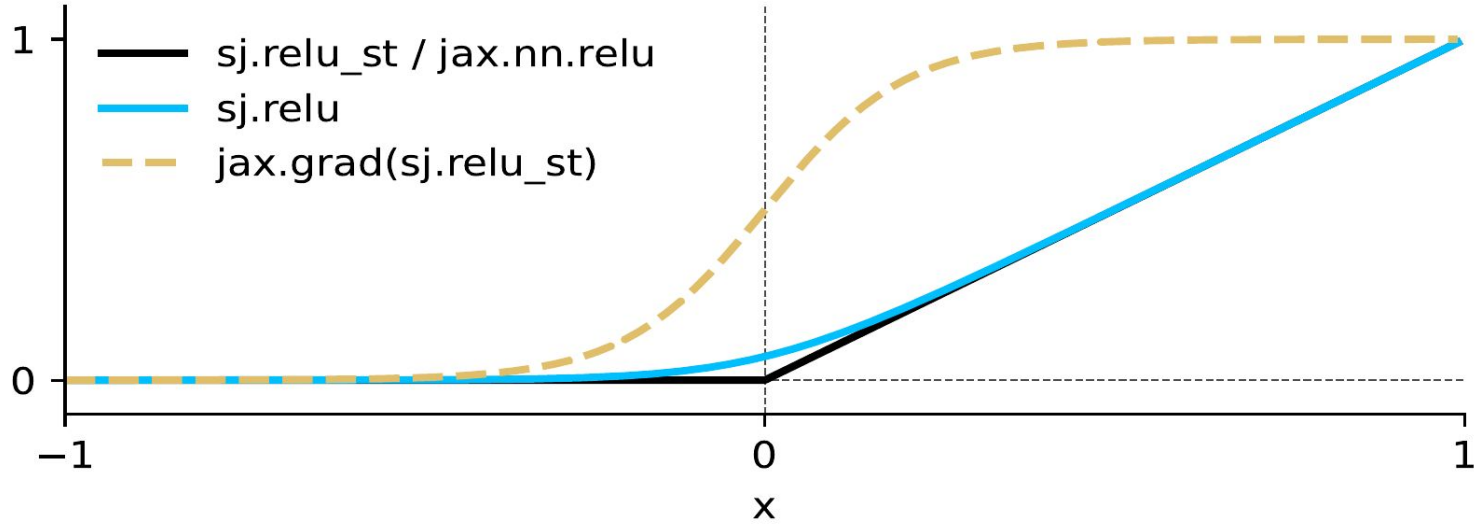
dynamic_slice

```
pip install softjax
```

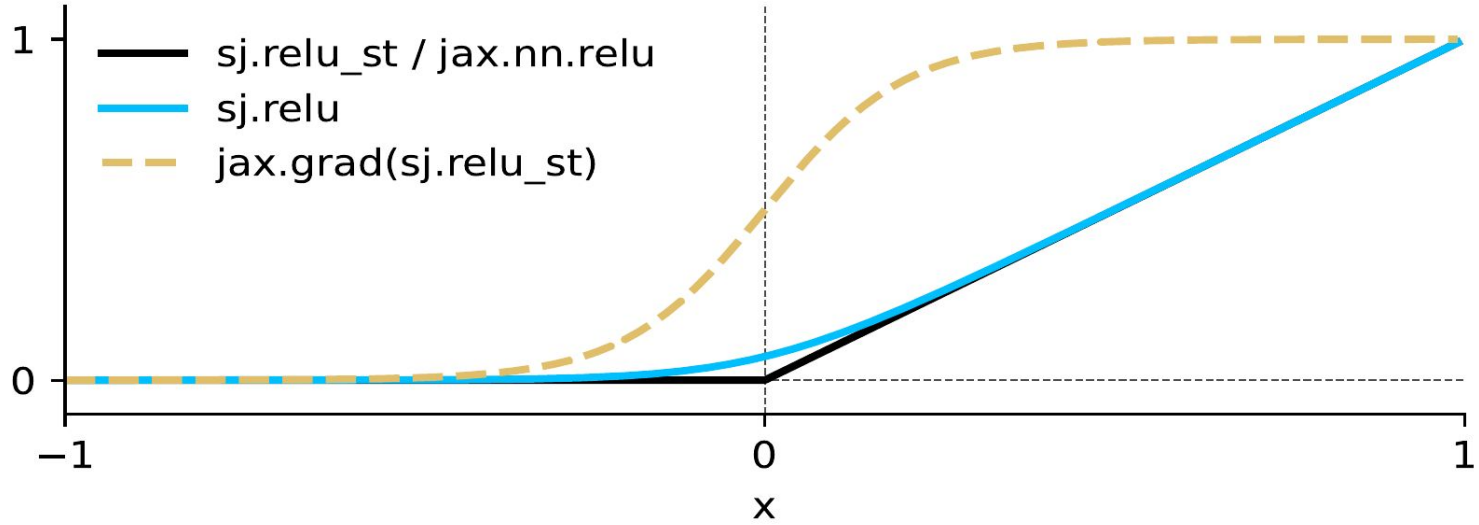
```
import softjax as sj
arr = jnp.array([x, -1.0, 0.5, 0.8, 2.0])
ranks = sj.rank(arr, method="neuralsort",
                mode="smooth")[0]
```

SoftJAX & SoftTorch, A Paulus, **AR Geist**, V Musil, S Hoffmann, O Beker, G Martius, 2026.

Softening?



Straight-through estimation



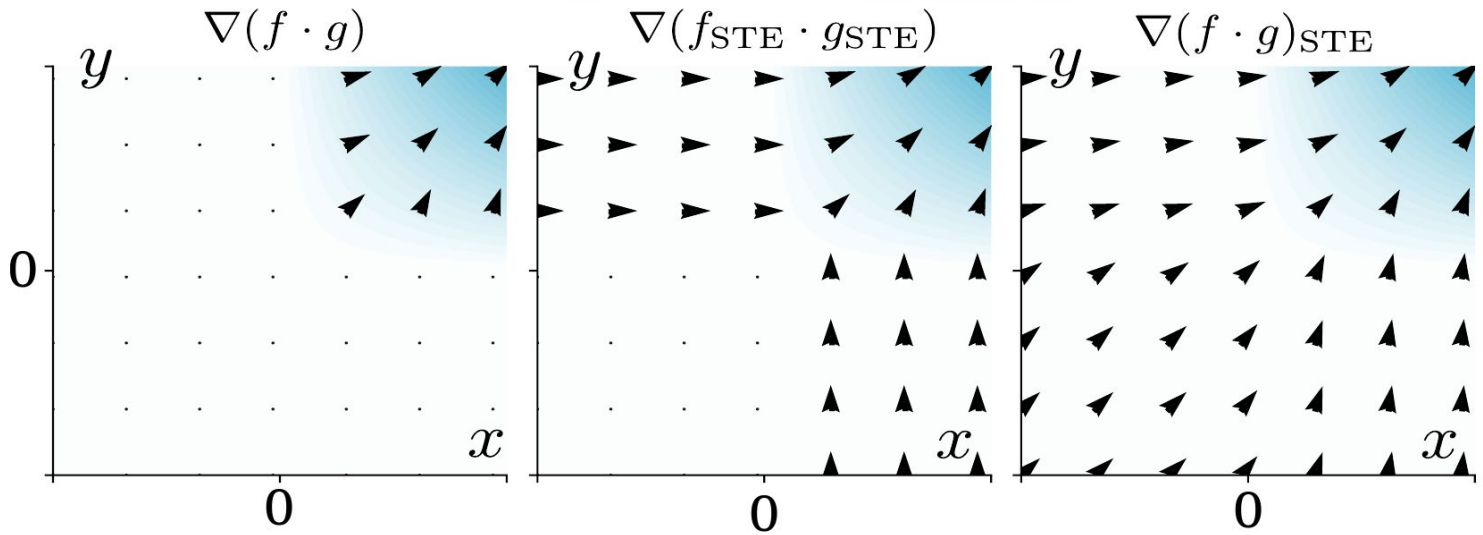
Straight-through trick

$$f_{\mathrm{STE}}(x) = \mathrm{sg}(f(x)) + f_\tau(x) - \mathrm{sg}(f_\tau(x))$$

Straight-through pitfall

$$\nabla(f_{\text{STE}} \cdot g_{\text{STE}}) = \nabla f_\tau \cdot g + f \cdot \nabla g_\tau$$

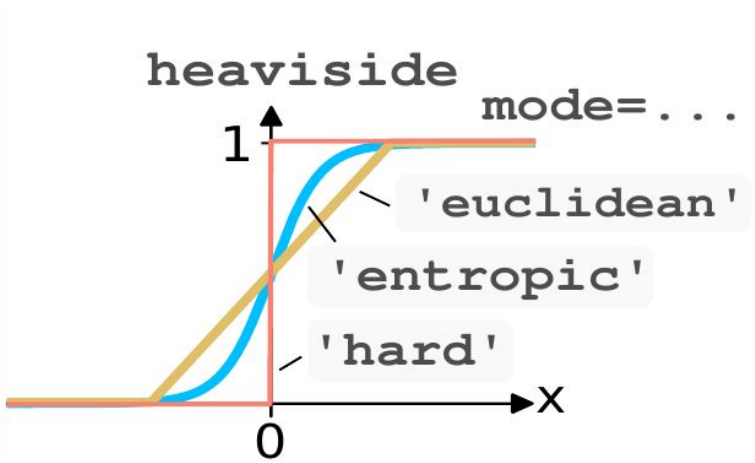
$$\nabla(f \cdot g)_{\text{STE}} = \nabla(f_\tau \cdot g_\tau) = \nabla f_\tau \cdot g_\tau + f_\tau \cdot \nabla g_\tau$$

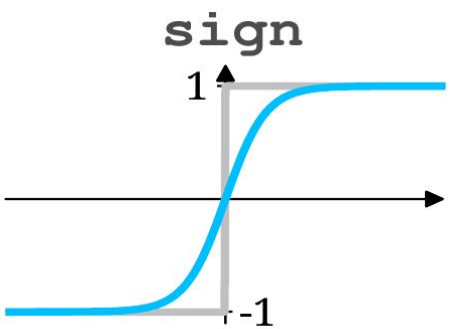


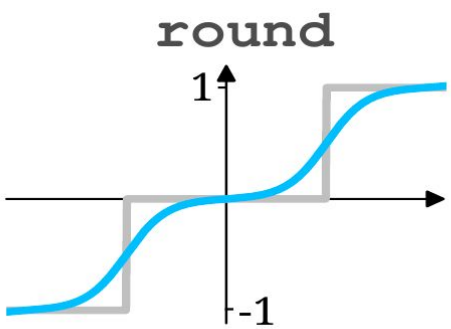
SoftJAX & SoftTorch, A Paulus, **AR Geist**, V Musil, S Hoffmann, O Beker, G Martius, 2026.

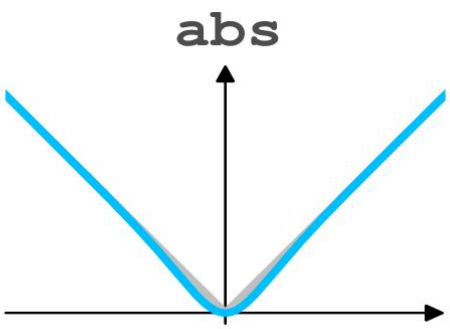
Elementwise operators

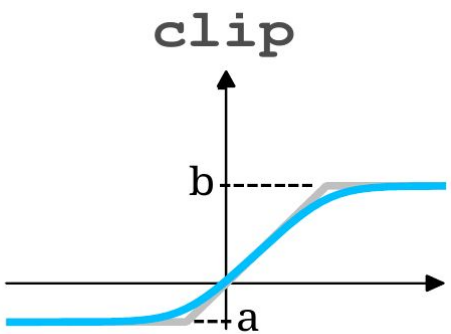
$$\mathrm{H}(x) := \begin{cases} 0 & \text{if } x < 0, \\ 0.5 & \text{if } x = 0, \\ 1 & \text{else.} \end{cases}$$





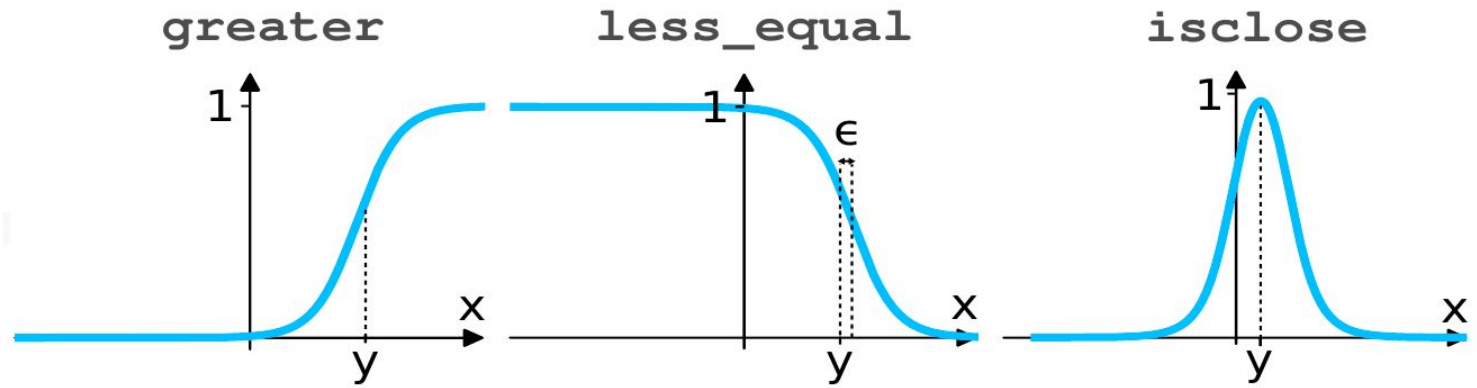






SoftJAX & SoftTorch, A Paulus, **AR Geist**, V Musil, S Hoffmann, O Beker, G Martius, 2026.

Comparisons



CDF defining the probability that a Boolean operator equates to true.

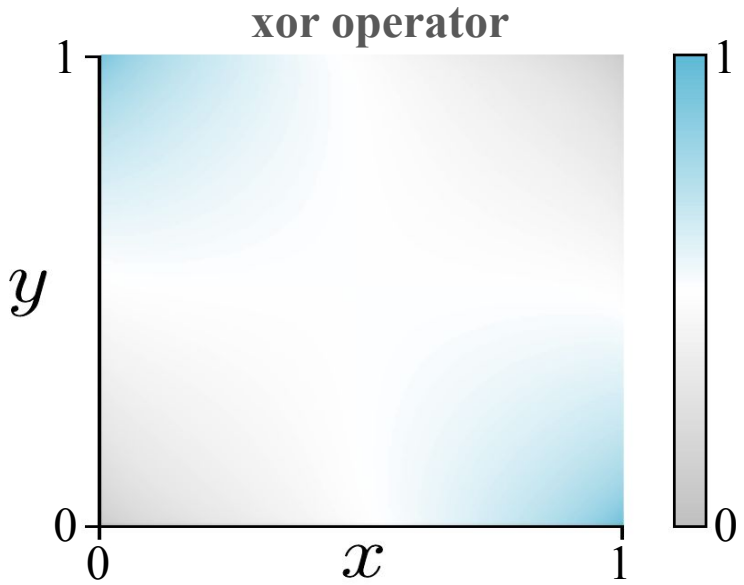
SoftJAX & SoftTorch, A Paulus, **AR Geist**, V Musil, S Hoffmann, O Beker, G Martius, 2026.

Fuzzy logic

$$\text{all}(p_1, \ldots, p_n) := \prod_{j=1}^{n} p_j$$

$$\text{not}(p) := \neg p = 1 - p$$

$$\begin{aligned}
\text{any}(p_1, \ldots, p_n) &:= \neg\, \text{all}(\neg p_1, \ldots, \neg p_n), \\
\text{and}(p, q) &:= \text{all}(p, q), \\
\text{or}(p, q) &:= \text{any}(p, q), \\
\text{xor}(p, q) &:= \text{or}\big(\text{and}(x, \neg y), \text{and}(\neg x, y)\big)
\end{aligned}$$



Selection = Expectation: $z_i = p_i \cdot x_i + (1 - p_i) \cdot y_i$

SoftJAX & SoftTorch, A Paulus, **AR Geist**, V Musil, S Hoffmann, O Beker, G Martius, 2026.

Axiswise operators

```
x = jnp.array([0.1, 0.4, 0.8])

idx = jnp.argmax(x) # 2
y = jax.lax.dynamic_index_in_dim(x, idx) # [0.8]
```

index

value at index

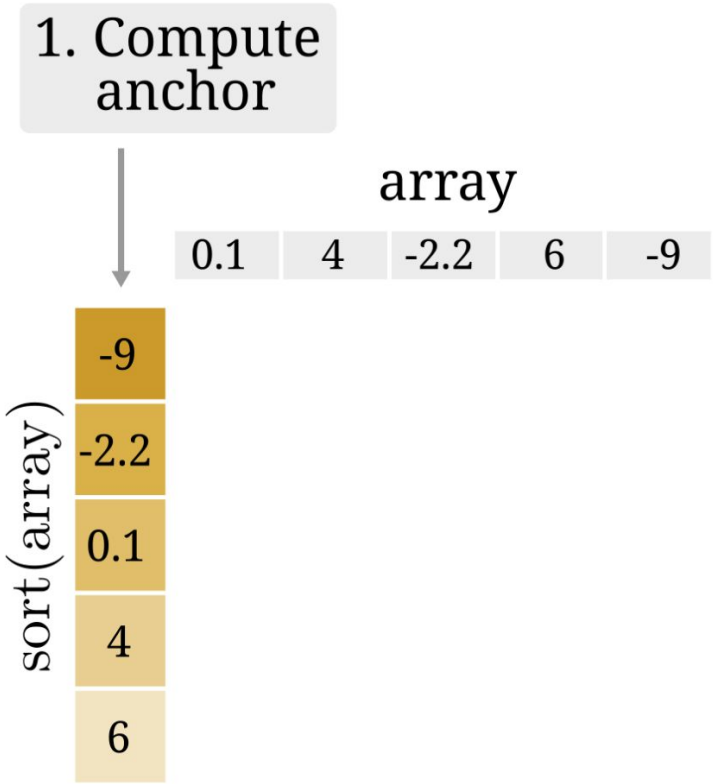
```
soft_idx = sj.argmax(x) # [0.004 0.042 0.953]
y = sj.dynamic_index_in_dim(x, soft_idx) # [0.78]
```

probability distribution over indices

expected value at index

SoftJAX & SoftTorch, A Paulus, **AR Geist**, V Musil, S Hoffmann, O Beker, G Martius, 2026.

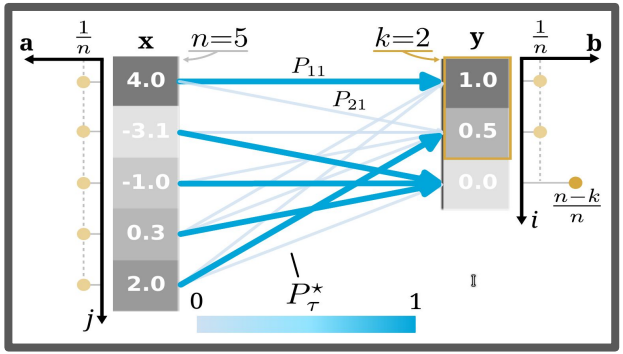
SoftSort



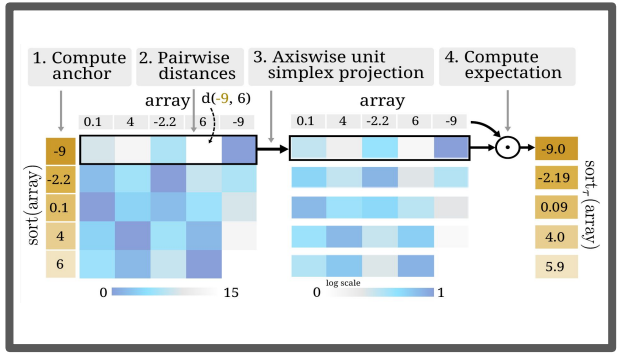
SoftJAX & SoftTorch, A Paulus, **AR Geist**, V Musil, S Hoffmann, O Beker, G Martius, 2026.

Axiswise operator overview

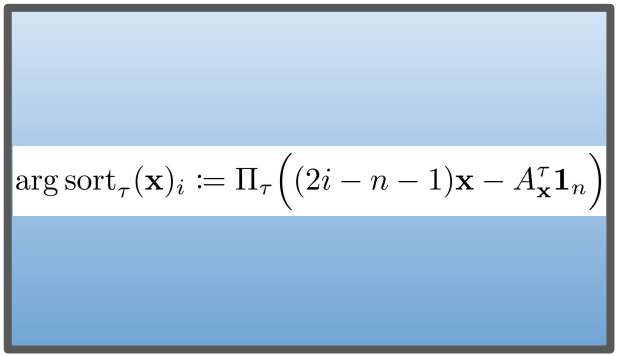
Regularized OT



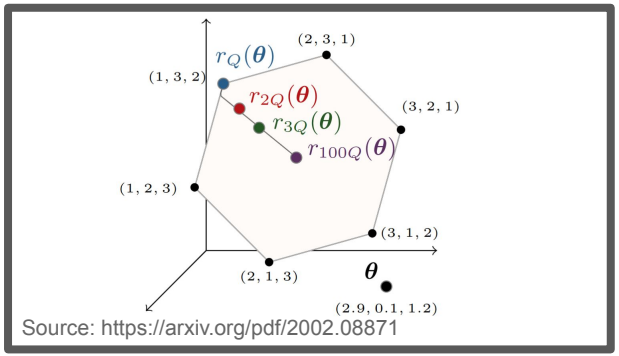
SoftSort



NeuralSort

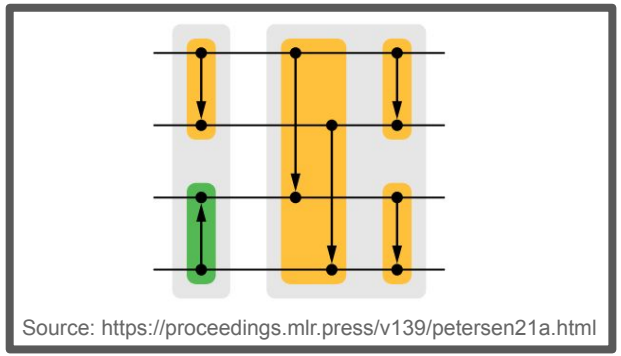


FastSoftSort



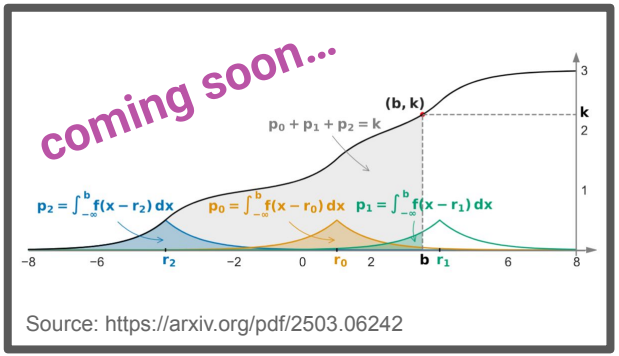
Source: https://arxiv.org/pdf/2002.08871

SortingNetworks



Source: https://proceedings.mlr.press/v139/petersen21a.html

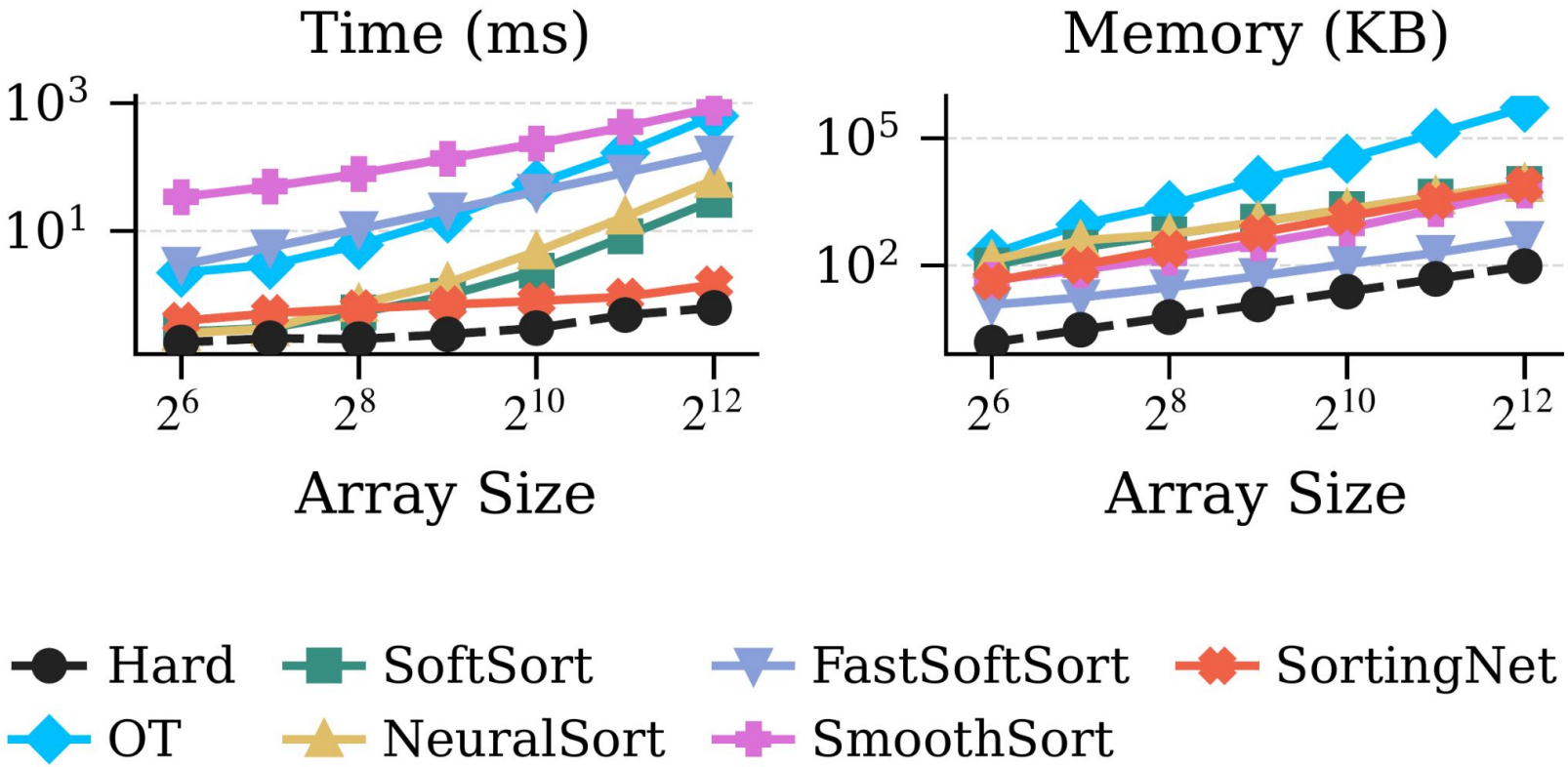
LapSum *and more...*



Source: https://arxiv.org/pdf/2503.06242

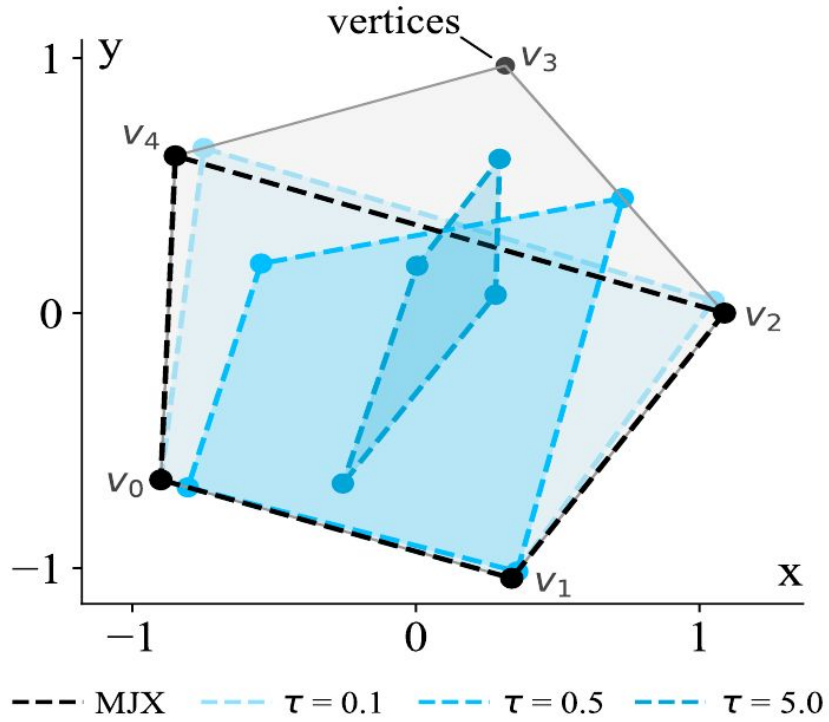
SoftJAX & SoftTorch, A Paulus, **AR Geist**, V Musil, S Hoffmann, O Beker, G Martius, 2026.

Sort operator



SoftJAX & SoftTorch, A Paulus, **AR Geist**, V Musil, S Hoffmann, O Beker, G Martius, 2026.

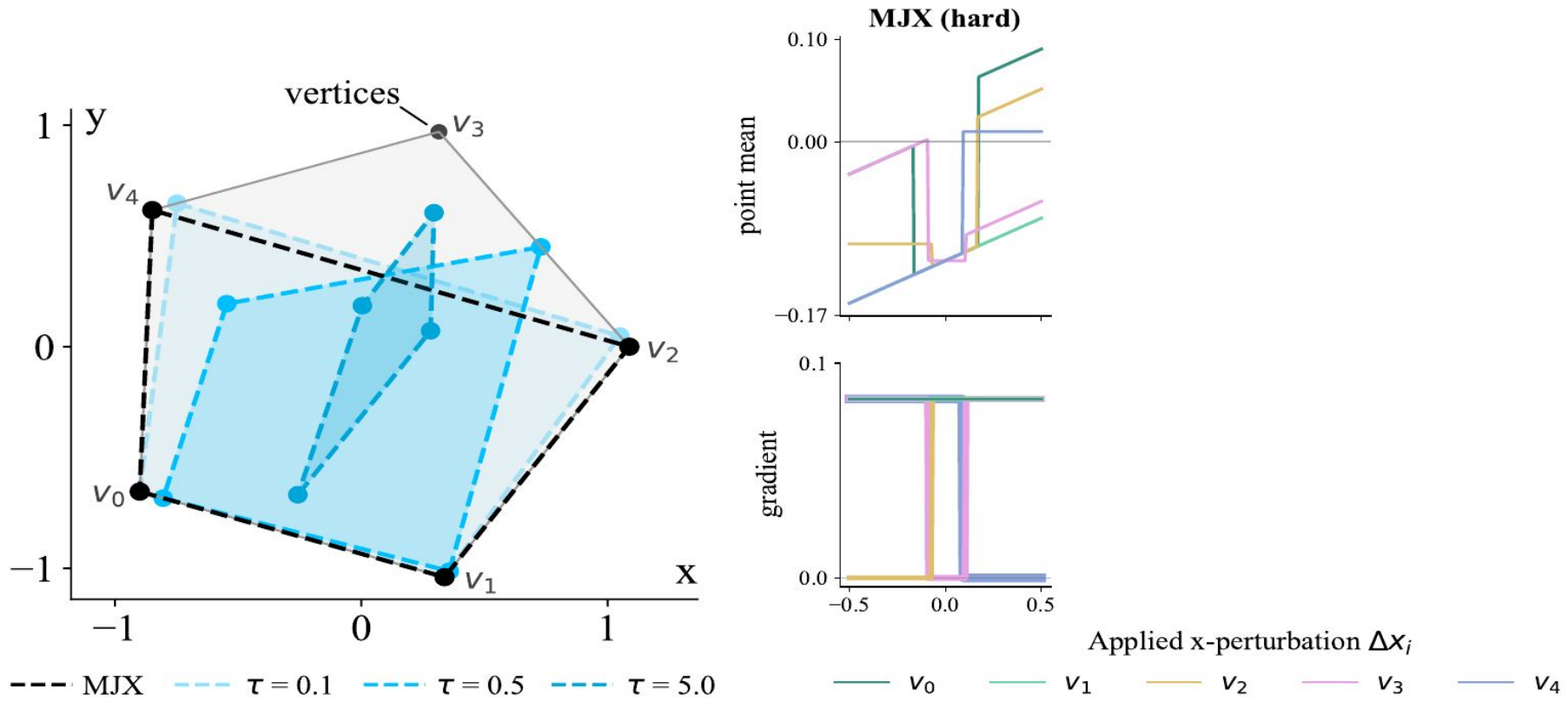
Case study - MuJoCo XLA




```
def manifold_points_mjx(poly, poly_mask, poly_norm:
     jnp.ndarray):

    dist_mask = jnp.where(poly_mask, 0.0, -1e6)
    # Note: We add a small tie-breaker
    # A: select the most penetrating vertex
    a_logits = dist_mask - 0.1 * jnp.arange(poly.shape
        [0])
    a_idx = jnp.argmax(a_logits)
    a = poly[a_idx]


    # B: farthest from A (largest squared distance)
    b_logits = ((a - poly) ** 2).sum(axis=1) +
        dist_mask
    b_idx = jnp.argmax(b_logits)
    b = poly[b_idx]


    # C: farthest from the AB line within the plane
    ab = jnp.cross(poly_norm, a - b)
    ap = a - poly
    c_logits = jnp.abs(ap.dot(ab)) + dist_mask

    c_idx = jnp.argmax(c_logits)
    c = poly[c_idx]
```

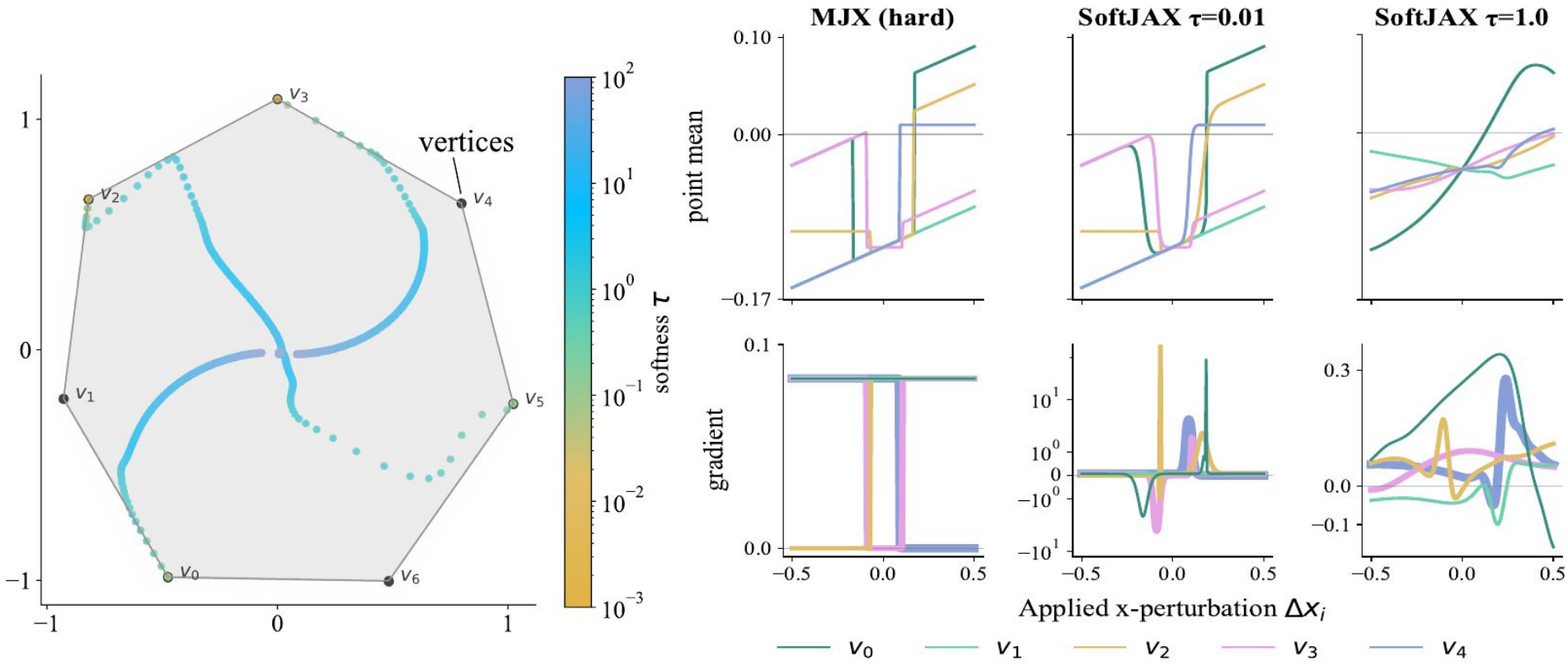
SoftJAX & SoftTorch, A Paulus, **AR Geist**, V Musil, S Hoffmann, O Beker, G Martius, 2026.

Case study - MuJoCo XLA



SoftJAX & SoftTorch, A Paulus, **AR Geist**, V Musil, S Hoffmann, O Beker, G Martius, 2026.

Case study - MuJoCo XLA



SoftJAX & SoftTorch, A Paulus, **AR Geist**, V Musil, S Hoffmann, O Beker, G Martius, 2026.

Thanks to my colleagues…

Anselm Paulus

Georg Martius

Vít Musil

Sebastian Hoffmann

Onur Beker

SoftJAX & SoftTorch: Empowering Automatic Differentiation Libraries with Informative Gradients,
A Paulus*, **AR Geist***, V Musil, S Hoffmann, O Beker, G Martius, 2026.