SciPy 1.15 will offer array API compatible functions for quadrature, series summation, differentiation, optimization, and root finding.

MATT HABERLAND[1,4], ALBERT STEPPI[2,4], AND PAMPHILE ROY[3,4]

1. CAL POLY, SAN LUIS OBISPO | 2. QUANSIGHT LABS | 3. CONSULTING MANAO GMBH | 4. SCIPY LIBRARY

Summary

Until recently, SciPy's best options for scalar quadrature, minimization, and root finding called compiled code, which could not take advantage of a vectorized Python integrand, objective function, or residual function; SciPy offered no functions for accurate numerical differentiation or series summation. These gaps are being filled with a family of pure-Python, array API compatible functions for dramatically faster vectorized calculation of scalar integrals, infinite sums, derivatives, minimizers, and roots.

Methods

Need: Differentiation of $f\colon \boldsymbol{R}^1 \to \boldsymbol{R}^1$
Solution: `scipy.differentiate.differentiate`

Need: Differentiation of $f\colon \boldsymbol{R}^m \to \boldsymbol{R}^n$
Solution: `scipy.differentiate.jacobian`

Need: Differentiation (2nd order) of $f\colon \boldsymbol{R}^m \to \boldsymbol{R}^1$
Solution: `scipy.differentiate.hessian`

Need: Integration of $f\colon \boldsymbol{R}^1 \to \boldsymbol{R}^1$
Solution: `scipy.integrate._tanhsinh`

Need: Summation of $\sum_{i=a}^{b} f(x_i)$
Solution: `scipy.integrate.nsum`

Need: Finding root of $f\colon \boldsymbol{R}^1 \to \boldsymbol{R}^1$
Solution: `scipy.optimize.elementwise.find_root`

Need: Bracketing root of $f\colon \boldsymbol{R}^1 \to \boldsymbol{R}^1$
Solution: `scipy.optimize.elementwise.bracket_root`

Need: Finding minimum of $f\colon \boldsymbol{R}^1 \to \boldsymbol{R}^1$
Solution: `scipy.optimize.elementwise.find_minimum`

Need: Bracketing minimum of $f\colon \boldsymbol{R}^1 \to \boldsymbol{R}^1$
Solution: `scipy.optimize.elementwise.bracket_minimum`

Acknowledgments

Thanks to everyone who has submitted feature requests, bug reports, and review comments related to these features.

This project has been made possible in part by grant EOSS-0000000432 from the Chan Zuckerberg Initiative DAF, an advised fund of the Silicon Valley Community Foundation.

References

[1] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., ... & Van Mulbregt, P. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature methods, 17(3), 261-272. https://doi.org/10.1038/s41592-019-0686-2.

Example Code

New sub-package for numerical differentiation

```
from scipy.differentiate import differentiate

def f(x, scale):
    return scipy.special.ndtr(x / scale)
```

Dispatches to array backend

Accepts objects compatible with Python array API standard

```
x = torch.asarray([[0.25],
                   [0.75]])

differentiate(f, x, args=(scale,))

scale = torch.asarray([1., 2., 3.])
```

Supports broadcasting; works elementwise

Example Result

```
success: tensor([[True, True, True],
                 [True, True, True]])
 status: tensor([[0, 0, 0],
                 [0, 0, 0]], dtype=torch.int32)
     df: tensor([[0.3867, 0.1979, 0.1325],
                 [0.3011, 0.1859, 0.1289]])
  error: tensor([[5.0664e-07, 1.5646e-06, 3.4422e-06],
                 [5.3644e-07, 1.5646e-06, 3.2485e-06]])
    nit: tensor([[2, 2, 2],
                 [2, 2, 2]], dtype=torch.int32)
   nfev: tensor([[11, 11, 11],
                 [11, 11, 11]], dtype=torch.int32)
```

Success/status information for each element

Accurate derivative and error estimates

Requires only a few (vectorized) calls to `f`