torchquad: Numerical Integration in Arbitrary Dimensions with PyTorch

Pablo Gómez¹, Håvard Hem Toftevaag¹, and Gabriele Meoni¹

1 Advanced Concepts Team, European Space Agency, Noordwijk, The Netherlands

Summary

torchquad is a Python module for n-dimensional numerical integration optimized for graphics processing units (GPUs). Various deterministic and stochastic integration methods, such as Newton–Cotes formulas and Monte Carlo integration methods like VEGAS Enhanced (Lepage, 2020), are available for computationally efficient integration for arbitrary dimensionality \( n_d \). As it is implemented using PyTorch (Paszke et al., 2019), one of the most popular machine learning frameworks, torchquad provides fully automatic differentiation throughout the integration, which is essential for many machine learning applications.

Statement of Need

Multidimensional integration is needed in many fields, such as physics (ranging from particle physics (Kersevan & Richter-Was, 2013) to astrophysics (Izzo & Gómez, 2021)), applied finance (Campolieti & Makarov, 2007), medical statistics (Ray et al., 2011), and machine learning (Atay & Hutt, 2006). Most of the conventional Python packages for multidimensional integration, such as quadpy (Schlömer et al., 2021) and nquad (Virtanen et al., 2020), only target and are optimized for central processing units (CPUs). However, as many numerical integration methods are embarrassingly parallel, GPUs can offer superior computational performance in their computation. Furthermore, numerical integration methods typically suffer from the so-called curse of dimensionality (Wu et al., 2020). This phenomenon refers to the fact that the computational complexity of the integration grows exponentially with the number of dimensions (Bellman, 2003). Reducing the error of the integration value requires increasing the number of function evaluation points \( N \) exponentially, which significantly increases the runtime of the computation, especially for higher dimensions. Previous work has demonstrated that this problem can be mitigated by leveraging the single instruction, multiple data parallelization of GPUs (Wu et al., 2020).

Although GPU-based implementations for multidimensional numerical integration in Python exist, some of these packages do not allow fully automatic differentiation (Borowka et al., 2019), which is crucial for many machine learning applications (Baydin et al., 2018). Recently, to fill this gap, the packages VegasFlow (Carraza & Cruz-Martinez, 2020) and ZMCintegral (Wu et al., 2020) were developed. Both of these implementations are, however, based on TensorFlow (Abadi et al., 2016), and there are currently no packages available that enable more than one-dimensional integration in PyTorch. Additionally, the available GPU-based Python packages that allow fully automatic differentiation rely solely on Monte Carlo methods (Carraza & Cruz-Martinez, 2020; Wu et al., 2020). Even though such methods offer good speed–accuracy trade-offs for problems of high dimensionality \( n_d \), the

*corresponding author
efficiency of deterministic methods, such as the Newton–Cotes formulas, is often superior for lower dimensionality (Lepage, 1978).

In summary, to the authors’ knowledge, torchquad is the first PyTorch-based module for \( n \)-dimensional numerical integration. Furthermore, it incorporates several deterministic and stochastic methods, including Newton–Cotes formulas and VEGAS Enhanced, which allow obtaining high-accuracy estimates for varying dimensionality at configurable computational cost as controlled by the maximum number of function evaluations \( N \). It is, to the authors’ knowledge, also the first GPU-capable implementation of VEGAS Enhanced (Lepage, 2020), which improves on its predecessor VEGAS by introducing an adaptive stratified sampling strategy.

Finally, being PyTorch-based, torchquad is fully differentiable, extending its applicability to use cases such as those in machine learning. In these applications, it is typically necessary to compute the gradient of some parameters with regard to input variables to perform updates of the trainable parameters in the machine learning model. With torchquad, e.g., the employed loss function can contain integrals without breaking the automatic differentiation required for training.

### Implemented Integration Methods

torchquad features fully vectorized implementations of various deterministic and stochastic methods to perform \( n \)-dimensional integration over cubical domains. In particular, the following deterministic integration methods are available in torchquad (version 0.2.1):

- Trapezoid Rule (Sag & Szekeres, 1964)
- Simpson's Rule (Sag & Szekeres, 1964)
- Boole's Rule (Ubale, 2012)

The stochastic integration methods implemented in torchquad so far are:

- Classic Monte Carlo Integrator (Caflisch, 1998)
- VEGAS Enhanced (VEGAS+) integration method (Lepage, 2020)

The functionality and the convergence of all the methods are ensured through automatic unit testing, which relies on an extensible set of different test functions. Both single and double precision are supported to allow different trade-offs between accuracy and memory utilization. Even though it is optimized for GPUs, torchquad can also be employed without a GPU without any functional limitations.

### Installation & Contribution

The torchquad package is implemented in Python 3.8 and is openly available under a GPL-3 license. Installation with either pip (PyPi)\(^1\) or conda\(^2\) is available. Our public GitHub repository\(^3\) provides users with direct access to the main development branch. Users wishing to contribute to torchquad can submit issues or pull requests to our GitHub repository following the contribution guidelines outlined there.

---

\(^1\)torchquad package on PyPi, https://pypi.org/project/torchquad/
\(^2\)torchquad package on conda, https://anaconda.org/conda-forge/torchquad
\(^3\)torchquad GitHub repository, https://github.com/esa/torchquad

---

Tutorials

The torchquad documentation, hosted on Read the Docs, provides some examples of the use of torchquad for one-dimensional and multidimensional integration utilizing a variety of the implemented methods.

References


Schlömer, N., Papior, N., Arnold, D., & Zetter, R. (2021). *Nschloe/quadpy* v0.16.6 (Version v0.16.6) [Computer software]. Zenodo. [Link](https://doi.org/10.5281/zenodo.4519699)


