Ускорение моделирования волоконных резонаторов на GPU при помощи CuPy и PyOpenCL

Ефремов Владислав Дмитриевич
младший научный сотрудник тем. группы 17-1 ИАиЭ СО РАН
Харенко Денис Сергеевич
к.ф.-м.н., ведущий научный сотрудник тем. группы 17-1 ИАиЭ СО РАН

Работа выполнена при финансовой поддержке госзадания ИАиЭ СО РАН (FWNG-2024-0015, рук. Бабин С.А., 2024-2026 гг.)

Аннотация

Библиотека для моделирования волоконных лазерных систем PyOFSS была адаптирована для расчётов на графических ускорителях при помощи библиотек CuPy и PyOpenCL. Было проведено сравнение точности и времени вычислений на основе аналитического решения в виде фундаментального солитона. Показано, что численные результаты дают приемлемую для моделирования высокую точность. Сравнение же времени расчётов показало, что CuPy справляется быстрее почти в 2 раза, чем PyOpenCL. С другой стороны, с помощью последней библиотеки моделирование генерации суперконтинуума в фотонно-кристаллическом волокне было выполнено в 2.5 раза быстрее. Сделан вывод, что со сложными расчётами PyOpenCL справляется лучше, однако за это приходится платить запутанной структурой кода, в противовес чему выступает универсальный и относительно простой CuPy, которая позволяет перенести расчёты всех элементов библиотеки, а те только оптического волокна.

1 Постановка задачи

Численное моделирование играет важную роль в современной науке. Оно позволяет рассматривать явления, которые трудно или вовсе невозможно исследовать экспериментально, а также даёт возможность предсказывать поведение сложных систем на основе предполагаемых параметров. В области волоконной оптики численное моделирование активно используется для исследования влияния нелинейных эффектов на оптические сигналы [1]. Это особенно важно при формировании сверхкоротких импульсов в волоконных лазерных резонаторах. Аналитических решений в таком случае не существует, а экспериментальное исследование затрудняется изобилием параметров схемы, вроде длины волокна, мощности накачки или поворота контроллера поляризации, а также вариаций компоновки отдельных элементов, например, расположением выводящего излучение из резонатора разветвителя. Таким образом, численное моделирование даёт возможность относительно быстро и без затрат материалов рассмотреть большое число различных схем и отобрать наиболее подходящую для практических задач. Кроме того, в последнее время набирает популярность способ обучения нейронных сетей на основе результатов численного моделирования для предсказания поведения излучения в экспериментальной установке [2]. Во всех случаях нужно повторять проход массива комплексной огибающей поля через все элементы резонатора в течении сотен циклов для каждого отдельного набора параметров схемы. Необходимость учитывать более сложные нелинейные эффекты, например, вынужденное комбинационное рассеяние, или дисперсию высшего порядка значительное усложняет вычисления. Как итог, одной из важных и всегда актуальных задач является ускорение моделирования волоконных резонаторов без потери точности.

2 Численное моделирование

Для моделирования волоконных лазерных резонаторов мы активно используем библиотеку Python-based optical fibre system simulator (PyOFSS). Язык программирования Python был выбран за относительную простоту команд и синтаксиса, большой набор готовых решений и специализированных библиотек (numpy, scipy и т.п.), заточенных на научных вычислениях, и поддержку объектно-ориентированного программирования. Благодаря этому у PyOFSS понятная структура, где каждому оптическому элементу в схеме отведён отдельный класс. Хорошо известно, что моделирование распространения излучения по волокну можно сделать быстрее при помощи графических ускорителей (Graphics processing unit, GPU), которые позволяют проводить математические операции над всеми точками массива одновременно. Ранее такой подход в PyOFSS был реализован при помощи библиотеки PyOpenCL, которая даёт полный контроль над GPU-вычислениями. Однако это имело два существенных недостатка. Во-первых, функции, которые производят операции с массивами на GPU, приходилось писать на особой версии языка C, что усложняло структуру. Также необходимо было контролировать буфер памяти и очередь команд. Во-вторых, это привело к появлению другой версии класса, отвечающего за моделирования волокна. Любые изменения, например, учёт дополнительных нелинейных эффектов, необходимо было дублировать в разном виде. Поэтому была предпринята попытка перейти на другую, более современную библиотеку для расчётов на GPU: CuPy. Её главная особенность заключается в практически полной схожести функций с NumPy и SciPy, которые активно используются для научных вычислений, в том числе, в PyOFSS. Таким образом, нет необходимости в дублировании классов, а в исходном коде библиотеки требуется минимум изменений. В общую структуру был лишь добавлен класс-синглтон (Singleton), через который остальные модули узнавали, какие из библиотек, NumPy с SciPy или CuPy, будет использоваться для расчётов.

Для сравнения скорости и точности вычислений использовалось аналитическое решение нелинейного уравнения Шрёдингера (НУШ). Фундаментальный солитон в виде:

$$A(t,z) = \sqrt{P_0}\,\mathrm{sech}\left(\frac{t}{T_0}\right)\exp\left(\frac{iz}{2L_D}\right), \tag{1}$$

где P_0 и T_0 – начальные мощность и длительность импульса, должен распространяться по волокну без изменений при совпадении дисперсионной $L_D = \frac{T_0^2}{|\beta_2|}$ и нелинейной $L_{NL} = \frac{1}{\gamma P_0}$ длин. Отсюда возникает условие на дисперсию второго порядка β_2 и керровскую нелинейность γ волокна. При правильно подобранных параметрах численное решение должно совпадать с аналитическим, на основе чего была проведена проверка скорости точности расчётов на GPU при помощи CuPy и PyOpenCL.

3 Результаты и обсуждение

Было проведено сравнение результатов после распространения солитона по волокну с длинами 1 и 100 км. Массив комплексной огибающей поля содержал 2^{18} точек. Для решения НУШ использовался метод Рунге-Кутты 4-го порядка в представлении взаимодействия. Вычисления были проведены на доступных GPU: на NVIDIA Tesla V100 SXM2 32GB, которая благодаря кластеру ИВЦ НГУ является основным расчётным инструментом, и на AMD Radeon RX 6700 XT, что стоит в лабораторном компьютере для локальных расчётов.

На Рис. 1 показано сравнение точности и скорости расчётов на GPU-NVIDIA для случая 1 км в зависимости от разного размера шага. Для примера также представлен результат, полученный при помощи стандартных библиотек NumPy и SciPy на CPU Intel Xeon X5670. Во всех случаях получается высокое совпадение с аналитическим решением, до 12-го знака после запятой при достаточном числе шагов. Сравнение же времени расчётов показало, что CuPy оказывается быстрее PyOpenCL более, чем в 2 раза. Расчёты на GPU-AMD дали тот же результат, несмотря на экспериментальную поддержку от CuPy графических ускорителей данного производителя. Для случая распространения фундаментального солитона по волокну длиной 100 км точность упала до 8-го и 9-го знаков после запятой для CuPy и PyOpenCL, соответственно, что является приемлемым результатом. Соотношение затраченного времени, однако, осталось прежним.

Для более сложных расчётов ситуация меняется кардинально. Было проведено моделирования генерации суперконтинуума в фотонно-кристаллическом волокне длиной 15 см [1]. Использовалось обобщённое НУШ, учитывающее дисперсию до 10-го порядка и эффект вынужденного комбинационного рассеяния на основе мультиколебательной модели. В результате расчёты на GPU-NVIDIA были произведены быстрее в 2.5 раза при помощи PyOpenCL, чем на CuPy. Как итог, можно сделать вывод, что с относительно простыми расчётами CuPy справляется быстрее из-за внутренней оптимизации часто используемых функций. Однако PyOpenCL даёт больший контроль над GPU-вычислениями, из-за чего корректно прописанная сложная функция будет выполняться быстрее, чем набор из большого числа сложения/умножения и функций типа numpy. За это приходится платить непрерывно растущей сложностью структуры кода в противовес простоте внедрения CuPy. Несмотря на всё, последнее видится более предпочтительным вариантом для дальнейшего расширения PyOFSS до моделирования многомодовых волокон.

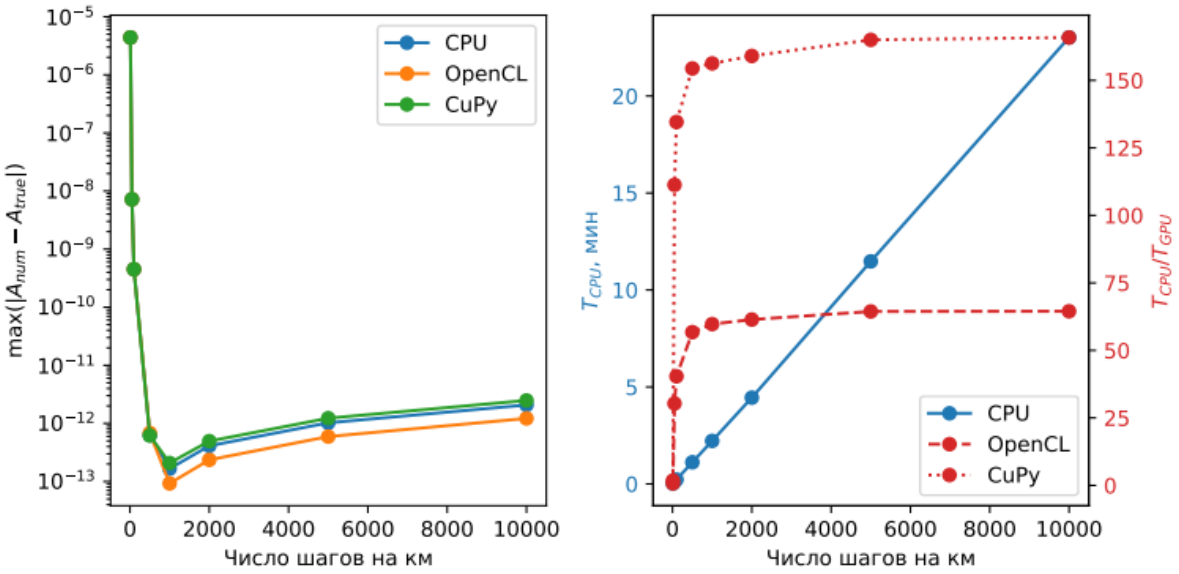

Рис. 1: Сравнение точности расчётов и потраченного на них времени при распространения фундаментального солитона по 1 км волокна в зависимости от размера шага. В качестве GPU использовалась NVIDIA Tesla V100 SXM2 32GB, CPU — Intel Xeon X5670.

4 Эффект от использования кластера

Кластер даёт доступ к 8-ми видеокартам NVIDIA Tesla V100 SXM2 32GB, что позволяет не только проводить большое число быстрых вычислений, но и постоянно улучшать способы расчётов сложных нелинейных эффектов в оптическом волокне. Без этого приходилось бы тратить в сотни раз больше времени на моделирование, что осложнило бы совместную работу с экспериментами, где порой требуется быстрый отклик на измеренные результаты, а также серьёзно ограничило бы диапазон варьируемых параметров при численном исследовании.

Литература

[1] Dudley J. M., Genty G., Coen S. Supercontinuum generation in photonic crystal fiber // Reviews of modern physics. 2006. Vol. 78. №. 4. P. 1135-1184. https://doi.org/10.1103/RevModPhys.78.1135

[2] Xuexiao Ma et al. Machine learning method for calculating mode-locking performance of linear cavity fiber lasers // Optics & Laser Technology. 2022. Vol. 149, № 107883. https://doi.org/10.1016/j.optlastec.2022.107883.