#### Применение многопоточного распараллеливания при использовании алгоритма Якоби

1. Основное отличие между последовательным и параллельным односторонним алгоритмом Якоби заключается в выборе стратегии выбора опорного элемента. В параллельных алгоритмах, в том числе на базе ЦПУ или ГПУ, стратегия опорного элемента выбирается таким образом, чтобы одновременно ортогонализовать максимальное количество независимых пар опорных элементов. Эти пары могут быть как двумя отдельными столбцами, так и блочными столбцами. Эта коллекция независимых пар будет называться шагом.

2. Модифицированная стратегия опорного элемента, которую мы выбрали как для алгоритмов на ЦПУ, так и для алгоритмов на ГПУ, является слегка модифицированной стратегией модуля. Если порядок матрицы четный, n=2q, то квазисвип выполняется точно с n шагами, и все элементы в верхнем треугольнике A аннигилируются (некоторые из них - дважды). Для q независимых задач (например, потоков ЦПУ или ГПУ) в каждом шаге одновременно ортогонализуется ровно q pivot пар.

3. Модифицированная модульная стратегия опорного элемента отличается выбором начального шага (наш - на антидиагонали) и аннигиляцией точно

Алгоритм GPUJACH1 SVD:

1. Initialize\_Stepper (Device\_Code (блок k))(r, SD=(ip, jp, i\_blk, j\_blk)):

   - Описание: Инициализация вектора SD стратегии модифицированного модулятора до антидиагонали - k↔(k, r−k−1).

   - Переменные ip и jp - вспомогательные, i\_blk и j\_blk - содержат индексы первого и второго столбца (не переставленные) пары опорных элементов для блока k.

   - Начало:

      - ipk = k;

      - i\_blkk = ipk;

      - jpk = r−k−1;

      - j\_blkk = jpk;

2. Update\_Stepper (Device\_Code (блок k))(r, SD=(ip, jp, i\_blk, j\_blk)):

   - Описание: Блок k определяет индексы следующей пары опорных элементов (i\_blkk, j\_blkk).

      - Если (ipk + jpk) > r−1, то

         - ipk = ipk + 1;

         - Если ipk = jpk, то

            - ipk = ipk−r/2;

            - jpk = ipk;

      - i\_blkk = ipk;

      - В противном случае,

         - jpk = jpk + 1;

         - j\_blkk = jpk;

3. Jacobi\_Step - основная вычислительная процедура на GPU:

3.1 Инициализация:

   - (i, j) = (SD(k).i\_blk, SD(k).j\_blk); // i < j

   - aii = di;

   - ajj = dj;

   - aij = gTρ(i)gρ(j); // скалярное произведение

3.2 Проверка ортогональности:

   - Если gρ(i) и gρ(j) относительно ортогональны (с точностью до ε), переходим к концу.

   - Определяем hyp = -1, если ji = jj, иначе hyp = 1.

   - Вычисляем угол ϕ:

     - Если hyp = -1, вычисляем t из aii, ajj, aij, как tan ϕ.

     - Если hyp = 1, вычисляем t как tanh ϕ.

   - Вычисляем cos и sinh от угла ϕ.

   - Выполняем FMA и масштабирование для обновления столбцов.

   - Обновляем матрицу D.

3.3 Обновление конвергенции:

   - Вызываем Update\_Convergence.

3.4 Завершение:

   - Вызываем Update\_Stepper для обновления индексов следующей пары pivot.

Этот алгоритм позволяет выполнить вычисления на GPU, при этом используя различные оптимизации для эффективного выполнения операций с матрицей

