МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**

ПРЕОБРАЗОВАНИЕ ФУРЬЕ. БЫСТРОЕ ПРЕОБРАЗОВАНИЕ ФУРЬЕ

ОТЧЕТ О ПРАКТИКЕ

Студента 3 курса 311 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Аношкина Андрея Алексеевича

Проверил
Старший преподаватель _____ М. С. Портенко

Саратов 2024

СОДЕРЖАНИЕ

1 Work 07

Задание

Проведите эксперименты для последовательного и параллельного вычислений БПФ, результаты занесите в таблицу. Оцените эффективность динамического планирования в OpenMP-версии по сравнению со статическим.

Быстрое преобразование Фурье

В работе изучаются общие идеи преобразования Фурье и алгоритма быстрого преобразования Фурье (БПФ) с последующей программной реализацией алгоритма БПФ в последовательном и параллельном случаях.

Общие сведения

Основу преобразования Фурье составляет идея о том, что почти любую периодическую функцию можно представить суммой гармонических составляющих или гармоник (синусоид с различными амплитудами, фазами и частотами).

Преобразование Фурье позволяет перейти от рассмотрения сигналов во временной области к их анализу и обработке в частотной области. Во временной области функция времени задается привычным образом, так как по оси абсцисс откладывается время. В частотной области функция времени отображается несколько иначе за счет того, что по оси абсцисс откладывается частота, а по оси ординат – амплитуда гармоник, составляющих функцию.

Представление функции в частотной области называют спектром функции.

Ряды Фурье

Пусть функция $f(t)$ представляет собой периодический сигнал, имеющий период T. Ряд Фурье функции $f(t)$ по ортогональной системе функций:

$$1, Cos\omega t, Sin\omega t, \ldots, Cosk\omega t, Sink\omega t$$

имеет вид:

$$f(t) = \frac{a_0}{2} + a_1 Cos\omega t + b_1 Sin\omega t + \cdots + a_k Cosk\omega t + b_k Sink\omega t + \ldots$$

♢ Основная частота $\omega = \frac{2\pi}{2}$ соответствует периоду T, остальные частоты кратны ей.

♢ Система функций ортогональна относительно скалярного произведения вида: $\int_{\alpha}^{\alpha+T} g(t)h(t)dt$

♢ $a_k = \frac{2}{T}\int_{\alpha}^{\alpha+T} f(t)Cosk \cdot \omega t dt$, $k = 0, 1, \ldots$

♢ $b_k = \frac{2}{T}\int_{\alpha}^{\alpha+T} f(t)Sink \cdot \omega t dt$, $k = 1, 2, \ldots$

Дискретное преобразование Фурье

♢ Рассмотрим выражение для комплексного коэффициента c_k:

$$\frac{1}{T}\int_{\alpha}^{\alpha+T} f(t) \cdot e^{-ik\omega t} dt,$$

где ω — основная частота.

♢ Выберем дискретные моменты времени для переведа задачи в дискретную форму.: $t_n = n \cdot \Delta t$, где Δt — период дискретизации.

♢ Выберем дискретные значения функции в эти моменты: $x_n = f(n \cdot \Delta t)$ (на полном периоде функции оказывается N точек: $N \cdot \Delta t = T$.

♢ Подставим полученные выражения в формулу для коэффициентов c_k.

Таким образом, интеграл аппроксимируется интегральной суммой (dt превратится в Δt, также принимается допущение, что за пределами сетки функция периодически повторяется):

$$c_k = \frac{1 \cdot \Delta t}{N \cdot \Delta t}\sum_{n=0}^{N-1} x_n e^{-\frac{2\pi \cdot i \cdot n \cdot \Delta t \cdot k}{N \cdot \Delta t}}, k = 0, \ldots, N-1.$$

Масштабный коэффициент $\frac{1}{N}$ не влияет на относительную величину c_k, поэтому указывать его не будем.

Обозначим относительную величину коэффициента c_k через $X(k)$ и получим выражение для ***дискретного преобразования Фурье***:

$$X(k) = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i n k}{N}}, k = 0, \ldots, N-1.$$

Дискретное преобразование Фурье (ДПФ) ставит в соответствие N отсчетам дискретного сигнала, N отсчетов дискретного спектра, при этом предполагается, что и сигнал, и спектр являются периодическими и анализируются на

одном периоде.

Быстрое преобразование Фурье

- ♢ Представленная выше формула для вычисления ДПФ требует значительных затрат. Трудоемкость такого алгоритма имеет порядок $O(N^2)$.
- ♢ В настоящее время существует целая серия оптимизированных алгоритмов расчета ДПФ, которые объединяют под общим названием ***быстрое преобразование Фурье (БПФ, FFT, Fast Fourier Transform)***.
- ♢ БПФ не является аппроксимацией ДПФ. Это в точности ДПФ, но с уменьшенным количеством арифметических операций.
- ♢ БПФ — это алгоритм эффективного вычисления ДПФ с трудоемкостью $O(\frac{N}{2}\log_2 N)$

Основные концепции БПФ

Пусть $W_N = e^{-\frac{2\pi i}{N}}$ — ***комплексный поворачивающий множитель***, который постоянен для заданного N, тогда выражение для ДПФ примет вид:

$$X(k) = \sum_{n=0}^{N-1} x_n W_N^{nk}, k = 0, \ldots, N-1.$$

Разделение исходной последовательности прореживанием по времени

Прореживание по времени заключается в разделении ДПФ на сумму двух ДПФ длиной $\frac{N}{2}$: одно формируется из компонентов с четными индексами $x_0, x_2, x_4, \ldots$, другое — из компонентов с нечетными индексами $x_1, x_3, x_5, \ldots$

$$X(k) = \sum_{n=0}^{N/2-1} x_{2n} W_N^{2nk} + \sum_{n=0}^{N/2-1} x_{2n+1} W_N^{(2n+1)k}$$
$$= \sum_{n=0}^{N/2-1} x_{2n} W_{N/2}^{2nk} + W_N^k \sum_{n=0}^{N/2-1} x_{2n+1} W_{N/2}^{2nk}, k = 0, \ldots, N-1.$$

Теперь заменим каждую полученную сумму на две суммы длиной $N/4$, в свою очередь состоящие из четных и нечетных слагаемых:

$$X(k) = \sum_{n=0}^{N/4-1} x_{4n} W_{N/4}^{nk} + W_{N/2}^{k} + \sum_{n=0}^{N/4-1} x_{4n+2} W_{N/4}^{2nk} +$$
$$W_N^k \left(\sum_{n=0}^{N/4-1} x_{4n+1} W_{N/4}^{nk} + W_{N/2}^{k} \sum_{n=0}^{N/4-1} x_{4n+3} W_{N/4}^{nk} \right), k = 0, \ldots, N-1.$$

И так далее рекурсивно разделяем вычисления на две части, для этого размер входных данных должен быть степенью двойки: $N = 2^q, q \in N$. Вычисления разбиваются до тех пор, пока не дойдем до одного элемента. Далее выполняем фиктивное ДПФ над одним элементом, так как для одного числа поворачивающий множитель вычислять не нужно и ДПФ над числом x есть само число x.

Процедура объединения

В результате математических преобразований получили, что два ДПФ четных и нечетных временных отсчетов входного сигнала можно объединить в ДПФ полной длины, если просуммировать отсчеты четной последовательности с произведением отсчетов нечетной последовательности входных сигналов на поворачивающий множитель. Количество операций умножения при этом значительно уменьшается по сравнению с прямым вычислением ДПФ.

Последовательная реализация

Фрагмент кода решения приведен ниже:

```
1  void SerialFFT(complex<double>* inputSignal, complex<double>* outputSignal, int size) {
2      BitReversing(inputSignal, outputSignal, size);
3      SerialFFTCalculation(outputSignal, size);
4  }
5
6  void ParallelFFTCalculation(complex<double>* signal, int size) {
7      int m = 0;
8      for (int tmp_size = size; tmp_size > 1; tmp_size /= 2, m++);
9
10     for (int p = 0; p < m; ++p) {
11         int butterflyOffset = 1 << (p + 1);
12         int butterflySize = butterflyOffset >> 1;
13         double coeff = PI / butterflySize;
14
15         #pragma omp parallel for schedule(static)
```

```
16              for (int i = 0; i < size / butterflyOffset; i++)
17                      for (int j = 0; j < butterflySize; j++)
18                              Butterfly(signal, complex<double>(cos(-j * coeff),
19                                      sin(-j * coeff)), j + i * butterflyOffset, butterflySize);
20      }
21  }
```

Параллельная реализация

Фрагмент кода решения приведен ниже:

```
1   void ParallelFFT(complex<double>* inputSignal, complex<double>* outputSignal, int size) {
2       BitReversing(inputSignal, outputSignal, size);
3       //ParallelBitReversing(inputSignal, outputSignal, size);
4       ParallelFFTCalculation(outputSignal, size);
5   }
6
7   void ParallelFFTCalculation(complex<double>* signal, int size) {
8       int m = 0;
9       for (int tmp_size = size; tmp_size > 1; tmp_size /= 2, m++);
10
11      for (int p = 0; p < m; ++p) {
12              int butterflyOffset = 1 << (p + 1);
13              int butterflySize = butterflyOffset >> 1;
14              double coeff = PI / butterflySize;
15
16              #pragma omp parallel for schedule(static)
17              for (int i = 0; i < size / butterflyOffset; i++)
18                      for (int j = 0; j < butterflySize; j++)
19                              Butterfly(signal, complex<double>(cos(-j * coeff),
20                                      sin(-j * coeff)), j + i * butterflyOffset, butterflySize);
21      }
22  }
```

Результат работы

Рисунок 1 – Work-7

Таблица сравнения

Номер теста	Размер входного сигнала	Последовательный алгоритм	Параллельный алгоритм	
			Время	Ускорение
1	32768	0.061	0.022	≈ 2.77
2	65536	0.145	0.046	≈ 3.15
3	131072	0.278	0.093	≈ 2.98
4	262144	0.59	0.196	≈ 3.01
5	524288	1.263	0.401	≈ 3.14

Использование динамического планирования замедляет работу.

Характеристики устройства

Процессор: Intel(R) Core(TM) i5-10400F

Ядер: 6

Оперативная память: 16 Гб