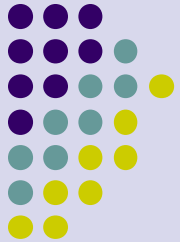
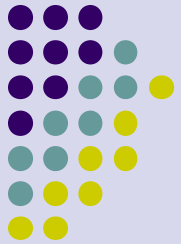
Летняя молодежная школа
“Разработка параллельных приложений для петафлопсных вычислительных систем”,
26 июня - 3 июля 2011 года,
НОЦ "Суперкомпьютерные технологии“,
МГУ имени М.В.Ломоносова



Разработка эффективных параллельных алгоритмов с использованием технологий Интел. Параллельные алгоритмы спектрального анализа

Панкратов Антон Николаевич

Особенности архитектуры, которые необходимо учитывать



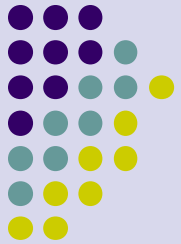
Кэш,

Конвейеризация вычислений,

Векторизация вычислений (Intel® IPP, MKL)

Многоядерность (OpenMP)

Обобщенный спектрально-аналитический метод



Вычисление коэффициентов разложения функции

Восстановление функции по коэффициентам разложения

Алгебра спектральных преобразований

Приложения

- Распознавание повторов в геномах и белках
- Аналитическое описание плоских и пространственных кривых, распознавание образов
- Обработка данных магнитной энцефалографии
- Обработка метеорологических данных
- Анализ данных ЯМР высокого разрешения, рентгеноструктурный анализ

Коэффициенты разложения

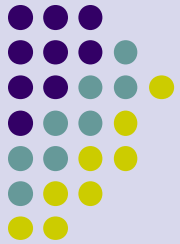
- Коэффициенты разложения по ортогональной на $[a;b]$ с весом $\rho(t)$ системе полиномов $\{u_k(t)\}$:

$$C_k = \int_a^b f(t)u_k(t)\rho(t)\,dt$$

- Определенные интегралы вычисляются с помощью соответствующих квадратурных формул Гаусса:

$$\int_a^b g(t)\rho(t)\,dt \approx \sum_{i=1}^{N} w_i g(t_i)$$

Как получать значения полиномов?



- Рекуррентное соотношение

$$U_{m+1}(t) = (A_m t + B_m)U_m(t) + U_{m-1}(t)$$

```
p2 = t;
p1 = 1;
u[0] = 1.0;
for (m = 1; m < N; m++) {
      p3 = p2;
      p2 = p1;
      p1 = 2*t*p2-p3;
      u[m] = p1;
}
```

Такой цикл не может быть распараллелен!

Сохранять ли значения полиномов?

Матричный подход

Если $A = (a_{ij})$ матрица значений полиномов:

$$a_{ij} = u_j(t_i), i = 1,..,N; j = 1,..M$$

то вычисление коэффициентов разложения

есть умножение матрицы на вектор-столбец

взвешенных значений:

где

$$\boxed{c = Af^t}$$

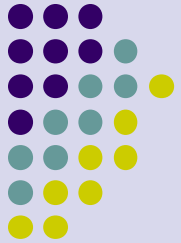
$$f = (f_1,..f_N);\ f_i = w_i f(t_i)$$

Матричный подход

+ вычисления выполняются заранее
+ возможность «включить» оператор интерполяции с одной сетки на другую
+ задача сводится к линейной алгебре

– интенсивное потребление памяти
– накладные расходы на доступ к памяти при работе на SMP-системах

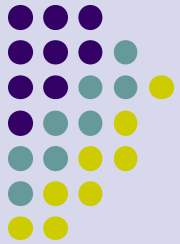
Вычисление значений полиномов «на месте»



+ экономия памяти
+ эффективное использование кэша
+ эффективное распараллеливание на SMP-системах

− повторяющиеся вычисления

Распараллеливание матричного алгоритма

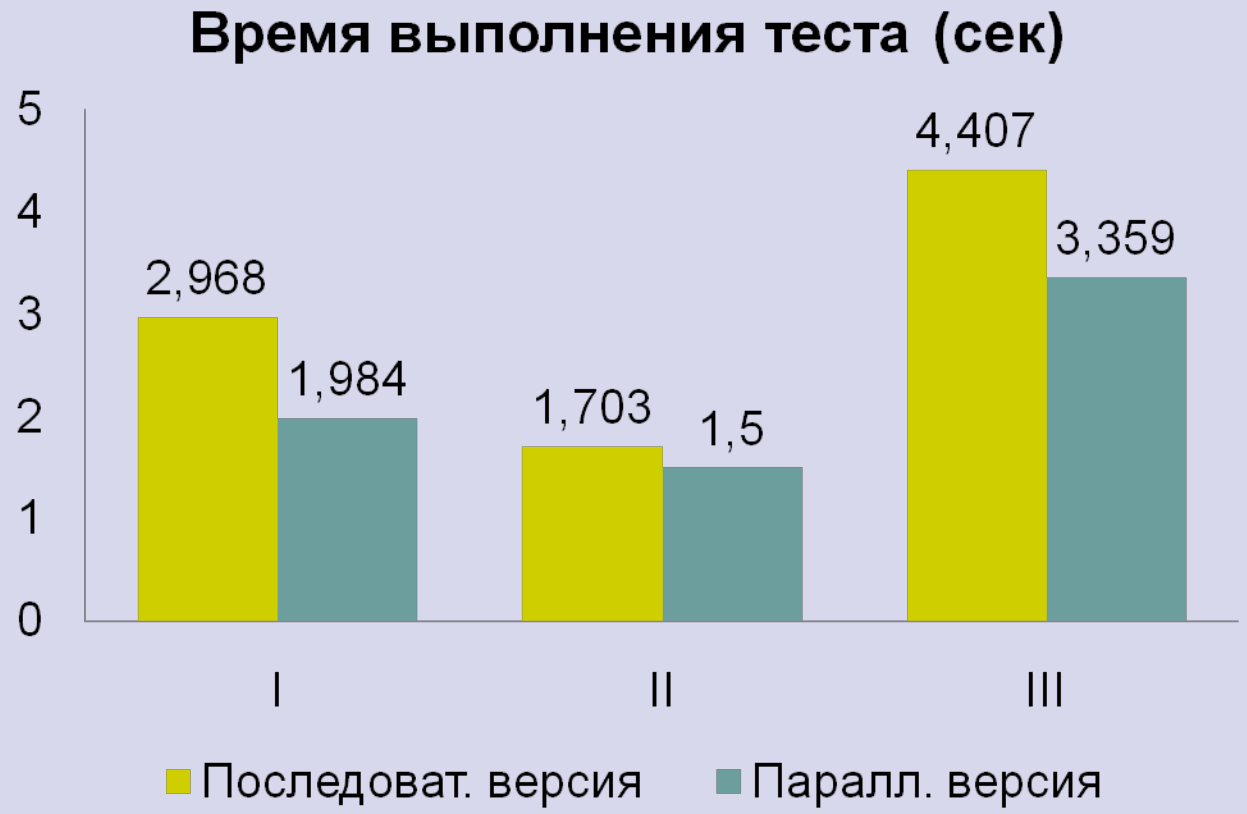


```
#pragma omp parallel for
for(k = 0; k < m; k++) {
   for(i = 0; i < n; i++) {
      c[k]+= f[i]*a[k*n+i];
   }
}
```

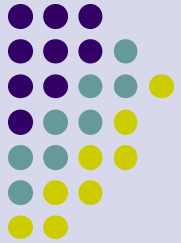
или в векторном виде (Matlab)

```
c=A*f’
```

Распараллеливание матричного алгоритма

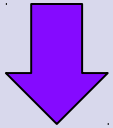


Распараллеливание «рекуррентного» алгоритма



```
#pragma omp parallel for reduction(c:+)
for(i = 0; i < n; i++) {
   p2 = 1.0;
   p1 = t[i];
   c[0] += y[i];
   for (j = 1; j < m; j++) {
       p3 = p2;
       p2 = p1;
       p1 = 2*t[i]*p2-p3;
       c[j] += y[i]*p1;
}
```

Каждый узел квадратурной сетки вносит вклад во все коэффициенты

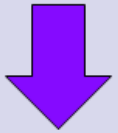


Внешний цикл нельзя распараллелить

Распараллеливание «рекуррентного» алгоритма

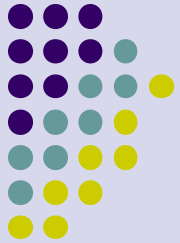
```
#pragma omp parallel for reduction(c:+)
for(i = 0; i < n; i++) {
    p2 = 1.0;
    p1 = t[i];
    c[0] += y[i];
    for (j = 1; j < m; j++) {
        p3 = p2;
        p2 = p1;
        p1 = 2*t[i]*p2-p3;
        c[j] += y[i]*p1;
}
```

Каждый узел квадратурной сетки вносит вклад во все коэффициенты



Внешний цикл нельзя распараллелить

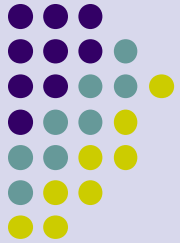
Распараллеливание «рекуррентного» алгоритма



```
#pragma omp parallel sections num_threads(2)
{
  #pragma omp section
  {
    cheb1_expand(t,y,N,0,N/2,c1,m);
  }
  #pragma omp section
  {
    cheb1_expand(t,y,N,N/2,N,c2,m);
  }
}
for (k = 0; k < m; k++) { c[k] = c1[k] + c2[k]; }
```

Распараллеливание «вручную» по узлам сетки

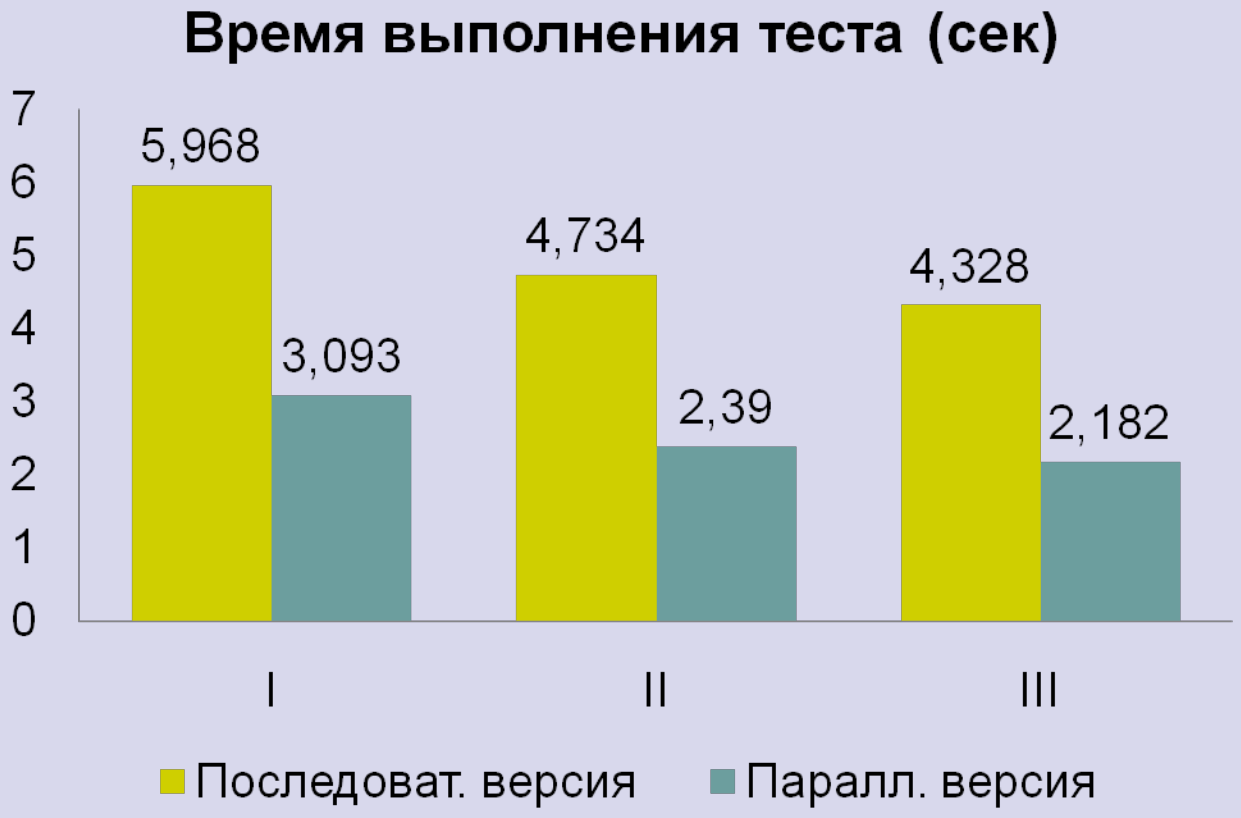
Векторизация «рекуррентного» алгоритма



```
for i = 1:k
  p1 = f;
  p2 = t.*f;
  c(1) = sum(p1);
  for j = 2:m
      p3 = p2;
      p2 = p1;
      p1 = 2*t.*p2-p3;
      c(j) = c(j) + sum(p1);
  end
end
```

Необязательный внешний цикл обусловлен дополнительной нарезкой сетки в случае фиксированной глубины векторизации

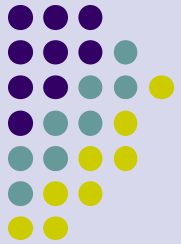
Распараллеливание «рекуррентного» алгоритма



Сравнение

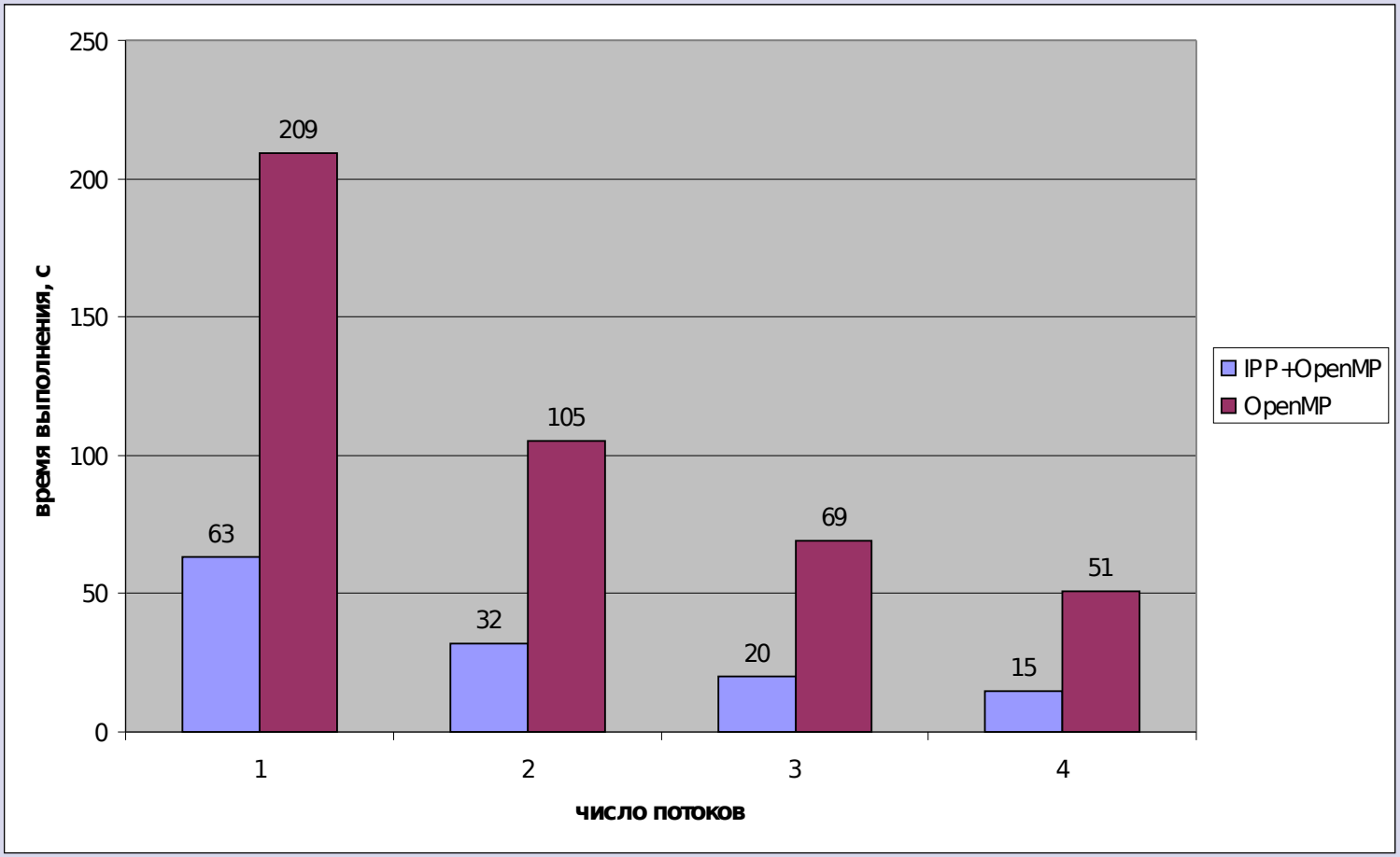
- Получены параллельные версии для процедур разложения функций по ортогональным полиномам
- Проведены сравнительные тесты матричного и «рекуррентного» алгоритмов на двухъядерных машинах
- Распараллеливание матричного алгоритма позволило не более, чем в 1,5 раза ускорить вычисления
- Распараллеливание «рекуррентного» алгоритма позволило в 2 раза ускорить вычисления
- Однако то, какой из алгоритмов: матричный или «рекуррентный» – эффективнее в абсолютном выражении, зависит от архитектуры ЭВМ

Алгоритмы вычисления коэффициентов разложения

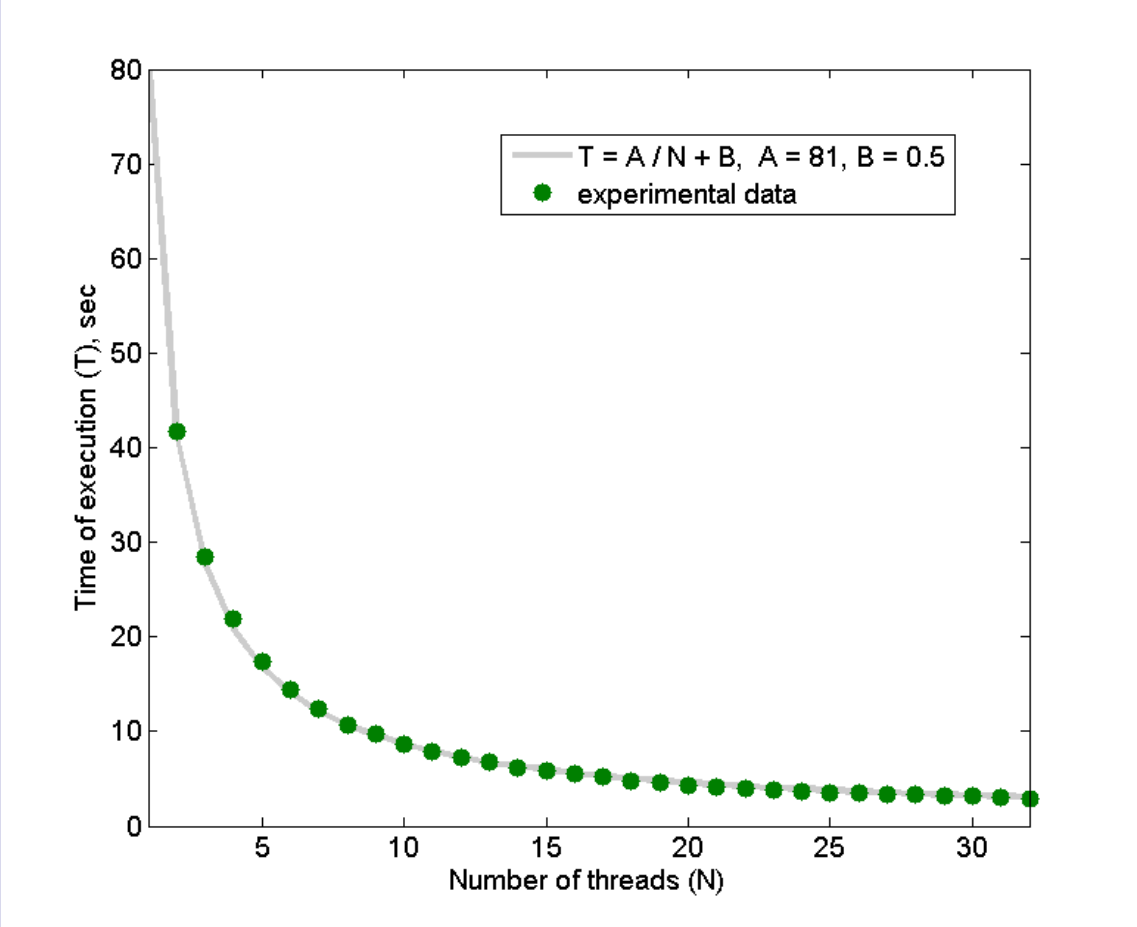


- Матричный
- Рекуррентный
- Векторно-рекуррентный
- Векторно-рекуррентный с фиксированной глубиной векторизации
- Быстрое преобразование Фурье (только тригонометрические функции, встроено практически во все библиотеки)

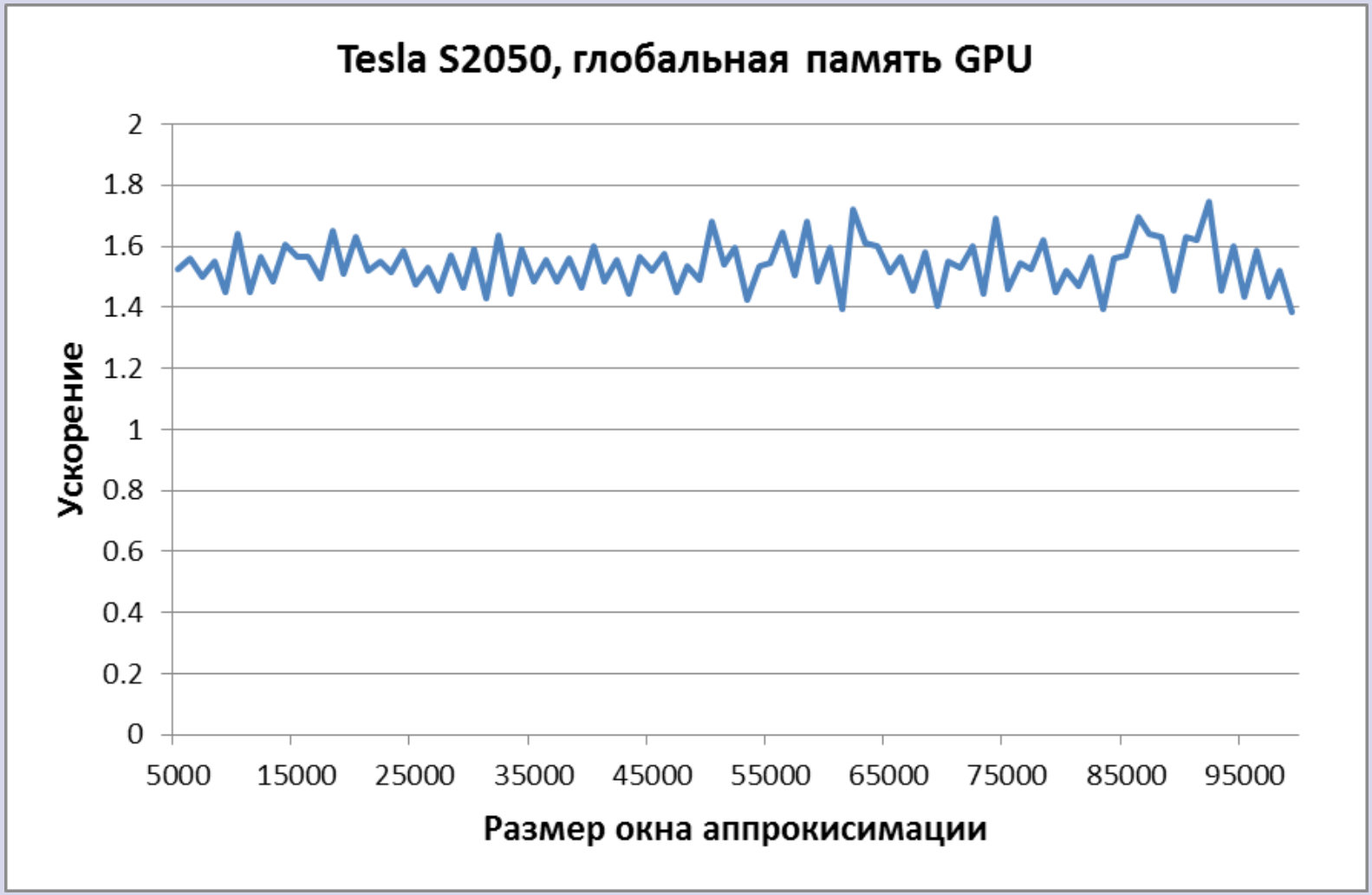
Использование векторизации и многоядерности



Intel MTL: 32-х ядерная архитектура

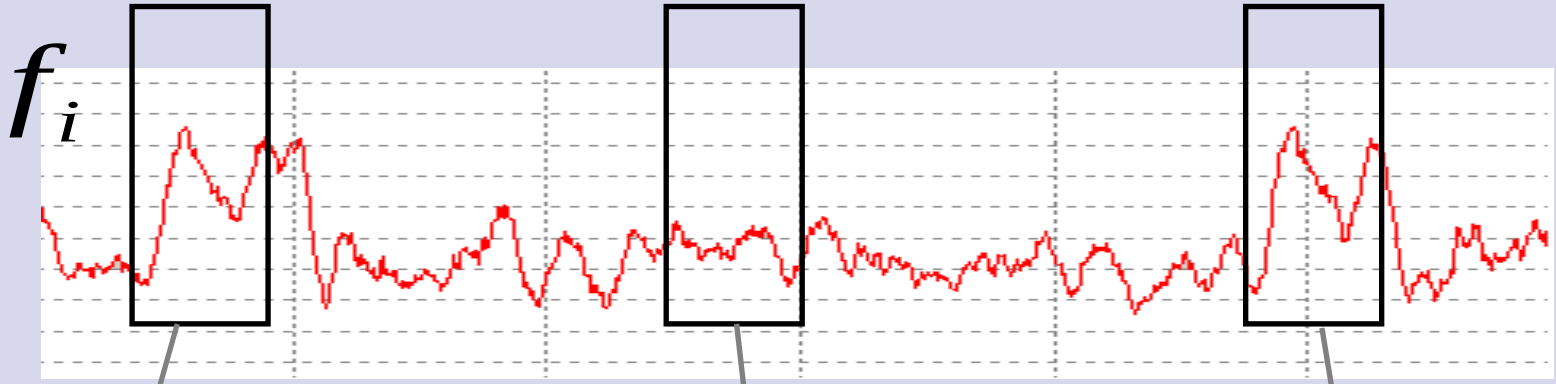


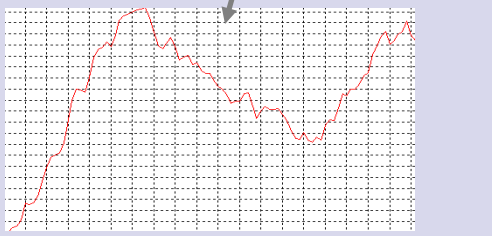
GPU vs CPU: однородность при изменении параметра

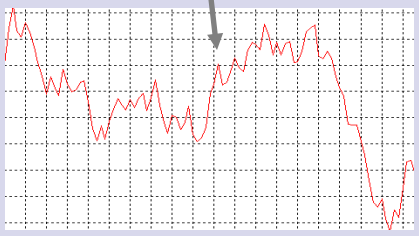


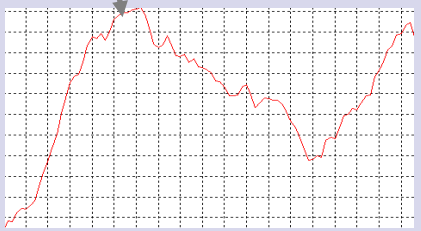
Поиск повторов в текстовых последовательностях

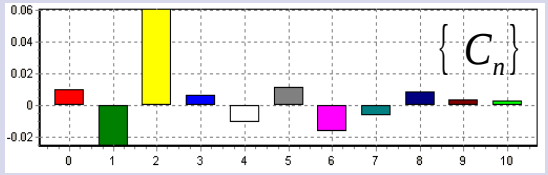
…ATGCGCATTCTCTGCCTGCATAAATCGCCGTATAAACCGCTACAATGCTACTGC…

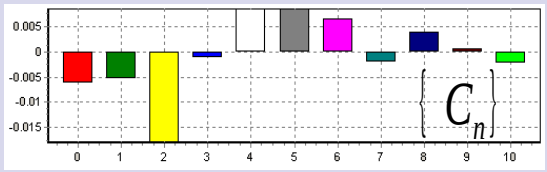


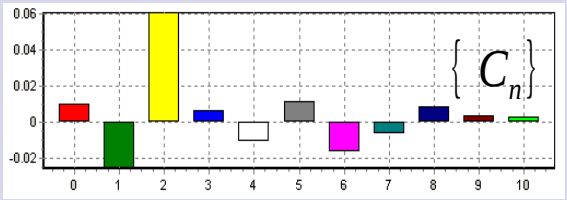








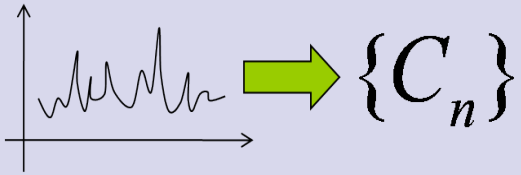




Алгоритм распознавания повторов

Перевод текстовой последовательности в непрерывную функцию

w1, окно частоты содержания букв



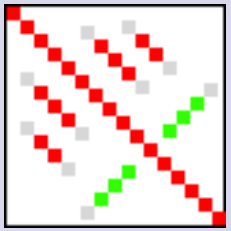
Спектральное индексирование последовательности

w2 , dw2
Окно и сдвиг окна аппроксимации

$$\theta(\{C_n\},\{C'_n\}) < \varepsilon$$

Построение решающего правила

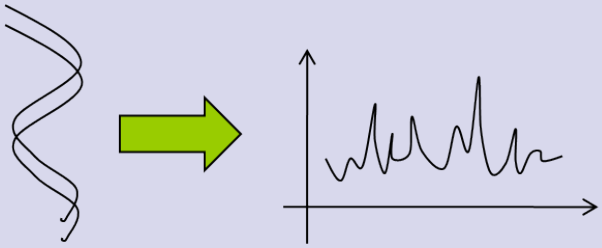
N, число коэффициентов разложения

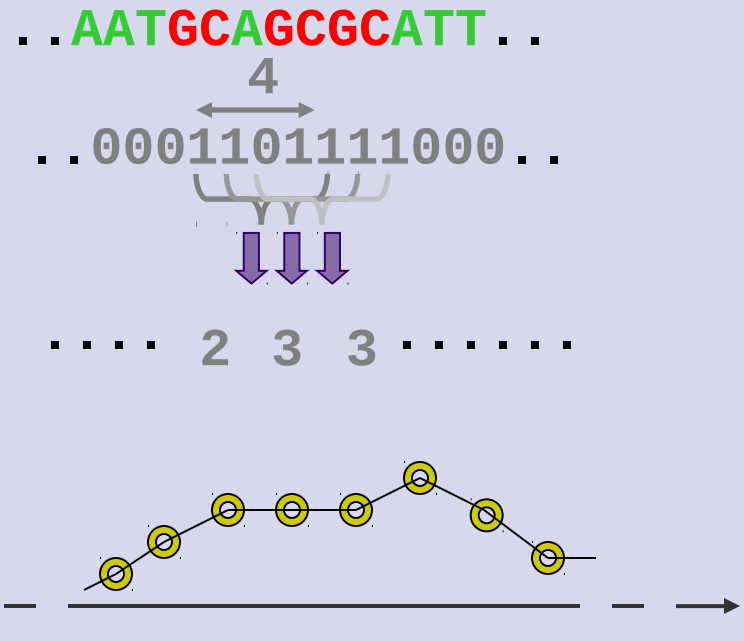


Отображение результатов на матрице спектральной гомологии

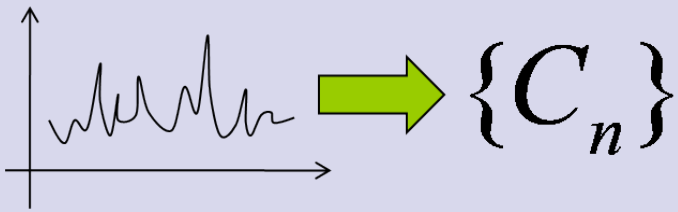
Шаг 1. Статистические профили текстовых последовательностей

Для однозначного представления текстовой последовательности в алфавите *N* букв требуется $Log_2 N$ профилей



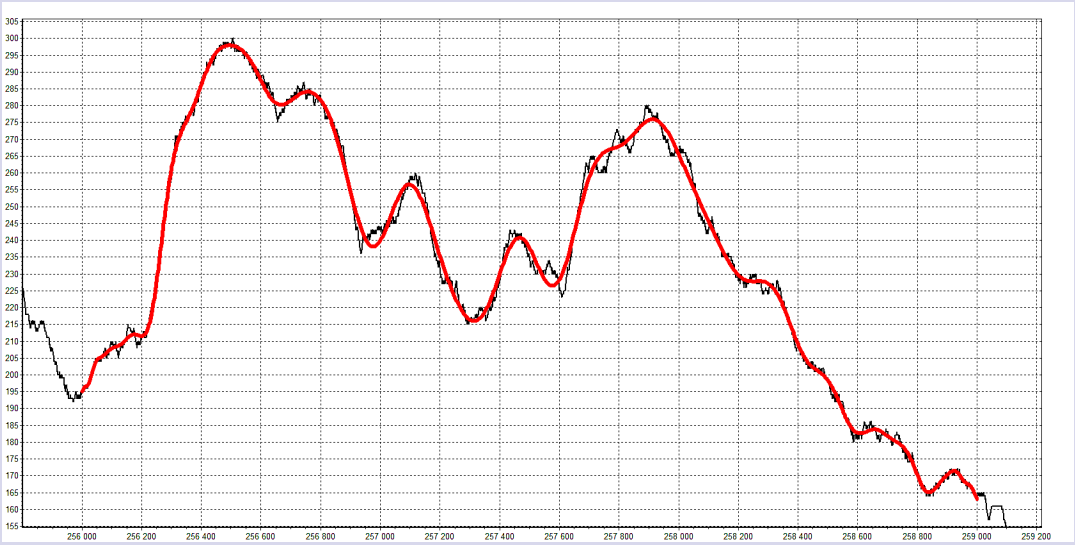


Шаг 2. Спектральное представление

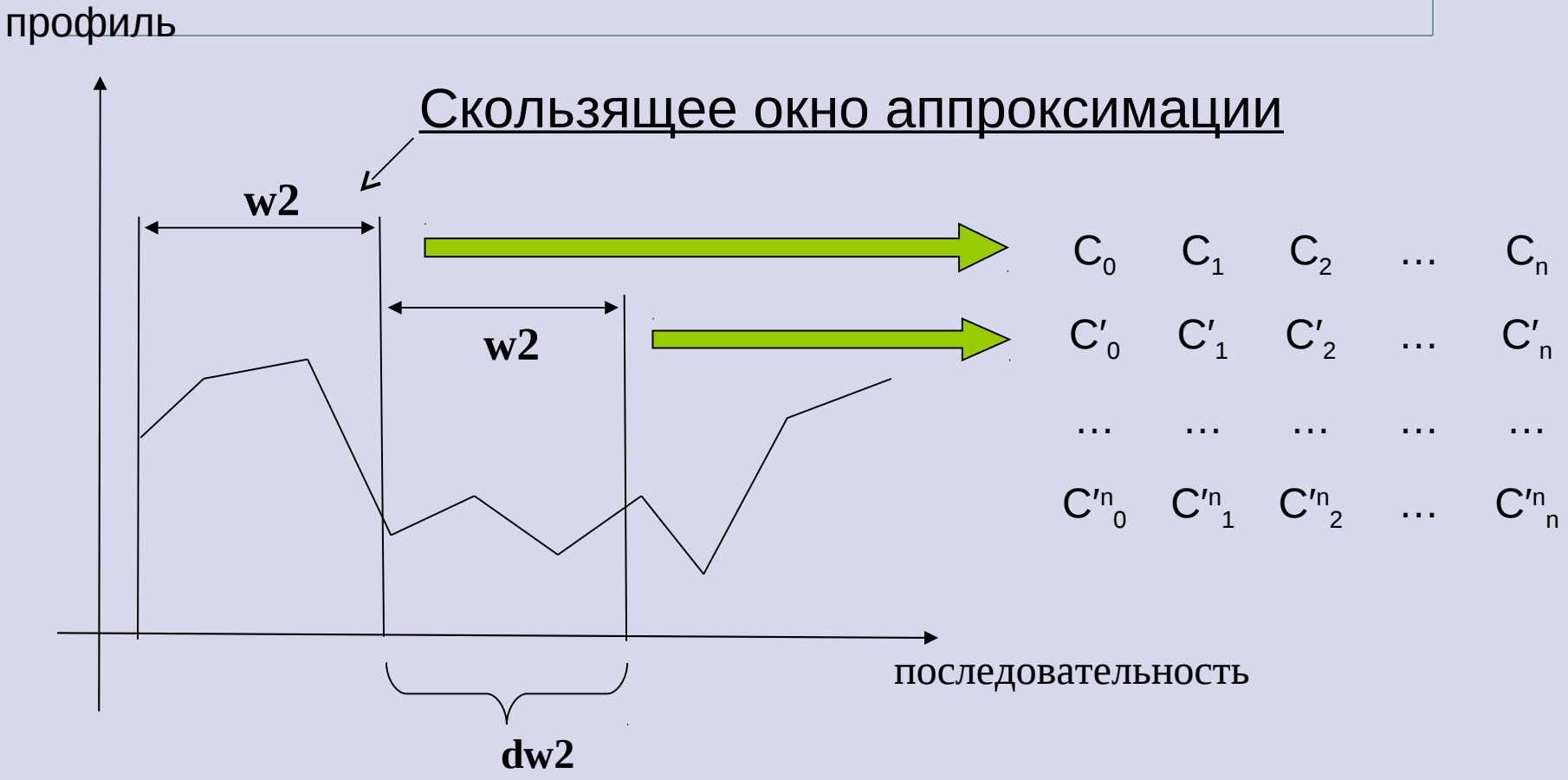


Разложение профилей содержания f_i по ортогональным функциям $\{\varphi_n\}$

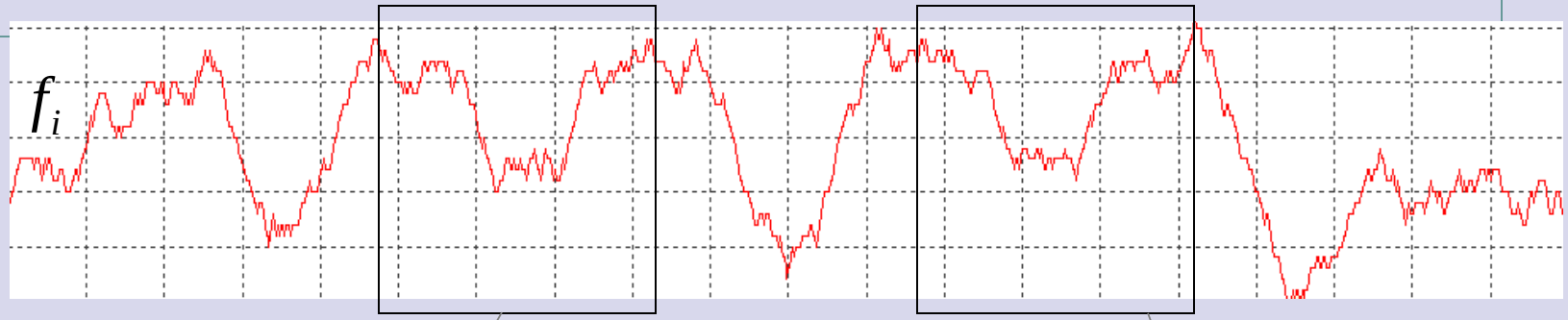
$$f_i^{(G,C)} \approx \sum_{n=0}^{N} C_n \varphi_n(x_i)$$

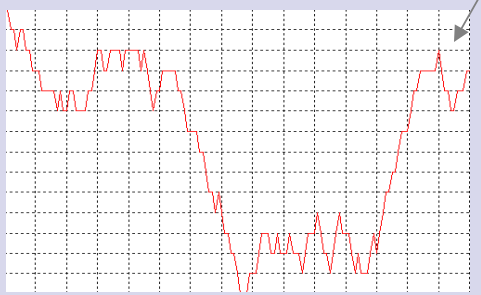


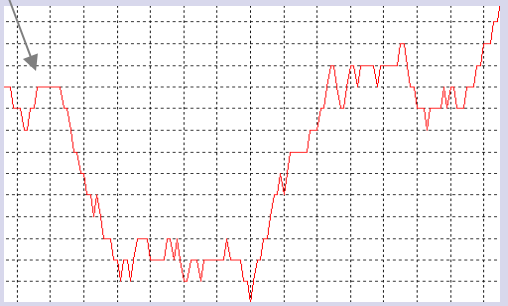
Спектральное индексирование



Поиск инвертированных повторов

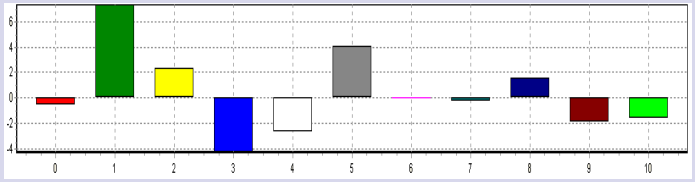






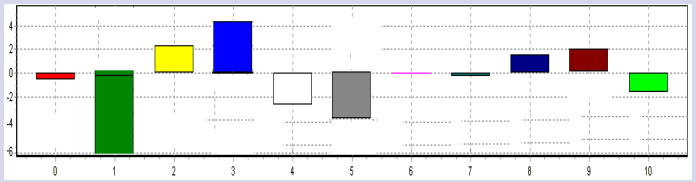
$$\begin{cases} \varphi_n(x) = \varphi_n(-x); n = 2k \\ \varphi_n(x) = -\varphi_n(-x); \quad n = 2k+1 \end{cases}$$

$$f^*(x) = f^{**}(-x)$$



$\{ C_n \}$

$$\begin{cases} C^*_{2k} = C^{**}_{2k} \\ C^*_{2k+1} = -C^{**}_{2k+1} \end{cases}$$

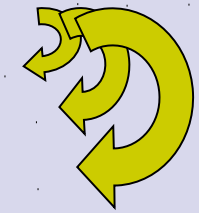


$\{ C_n \}$

Шаг 3. Определение решающего правила

$$\theta(\{C_n\},\{C'_n\}) < \varepsilon$$

C_0	C_1	C_2	…	C_n
C'_0	C'_1	C'_2	…	C'_n
…	…	…	…	…
C'^n_0	C'^n_1	C'^n_2	…	C'^n_n



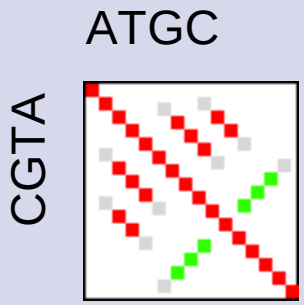
$$\theta = \frac{\| f - g \|}{\| f \| + \| g \|} \qquad 0 \le \theta \le 1$$

$$\theta_N = \frac{\| \sum_{n=0}^{N} A_n \varphi_n - \sum_{n=0}^{N} B_n \varphi_n \|}{\| \sum_{n=0}^{N_{\max}} A_n \varphi_n \| + \| \sum_{n=0}^{N_{\max}} B_n \varphi_n \|}$$

$$\theta_N \le \theta_{N+1} < \varepsilon$$

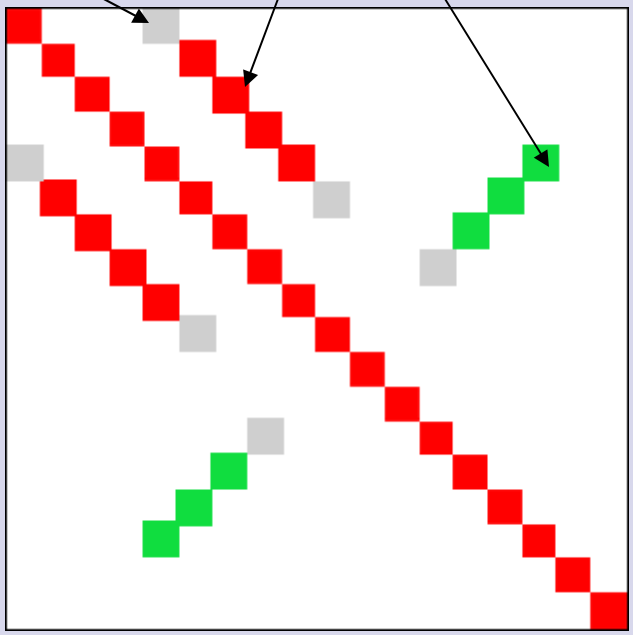
Метрика монотонна по числу коэффициентов

Шаг 4. Построение матрицы спектральной гомологии

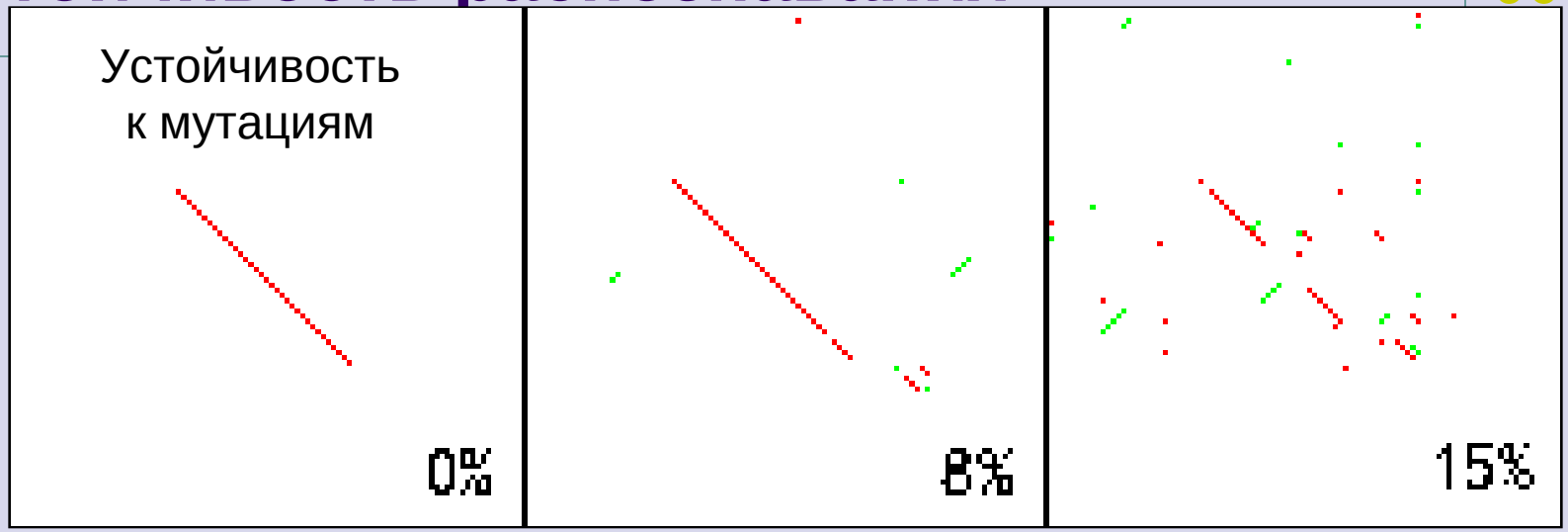


$\theta_{N\max} < \varepsilon$

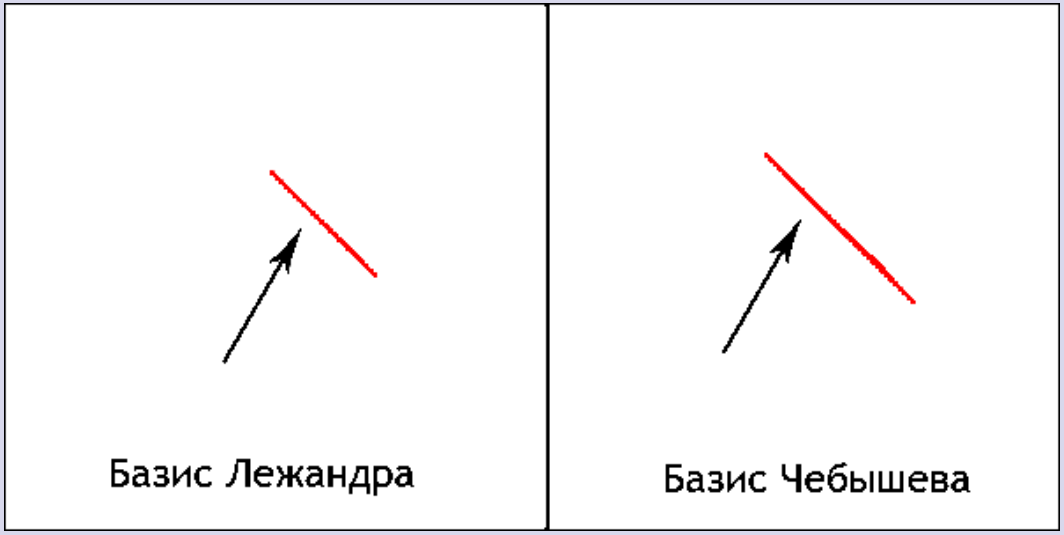
$\theta_N > \varepsilon$



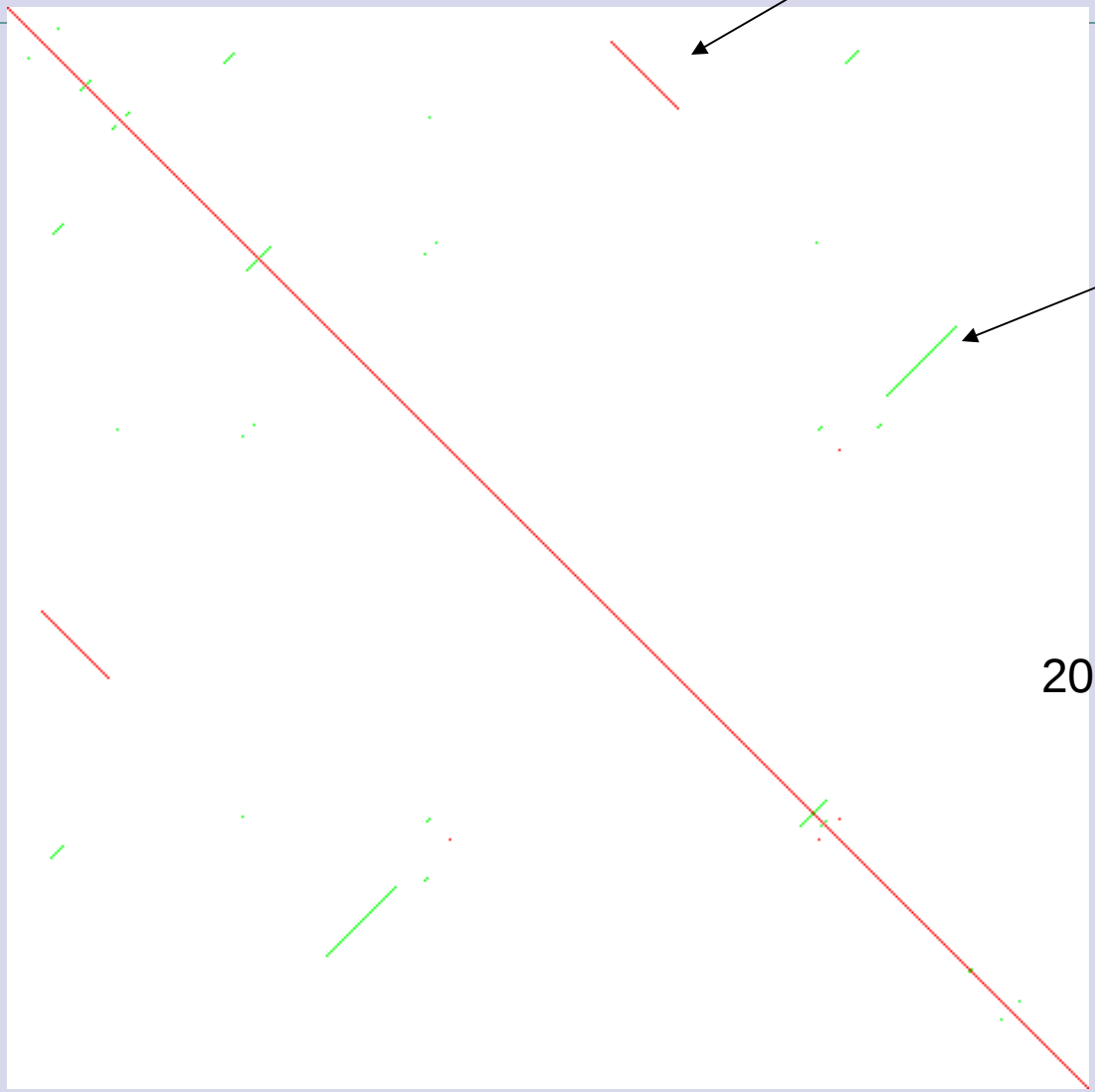
Устойчивость распознавания



Выбор базиса

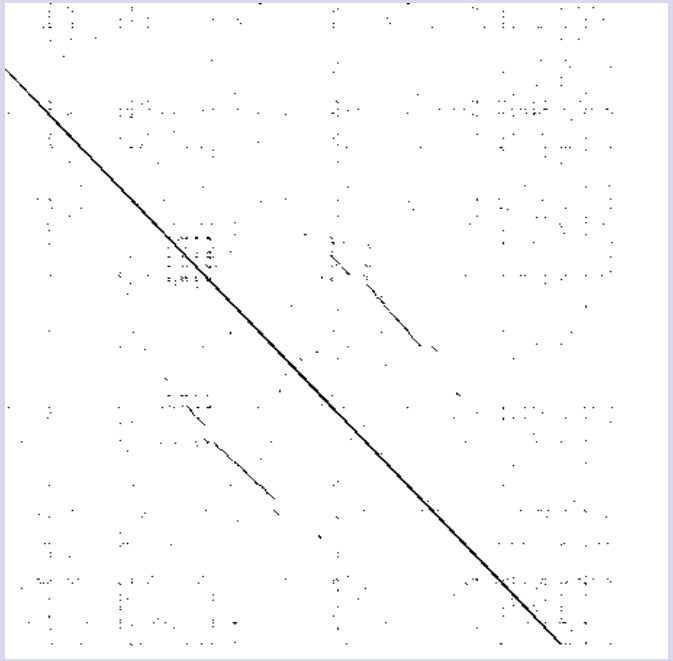


Обучение алгоритма на тестовой последовательности

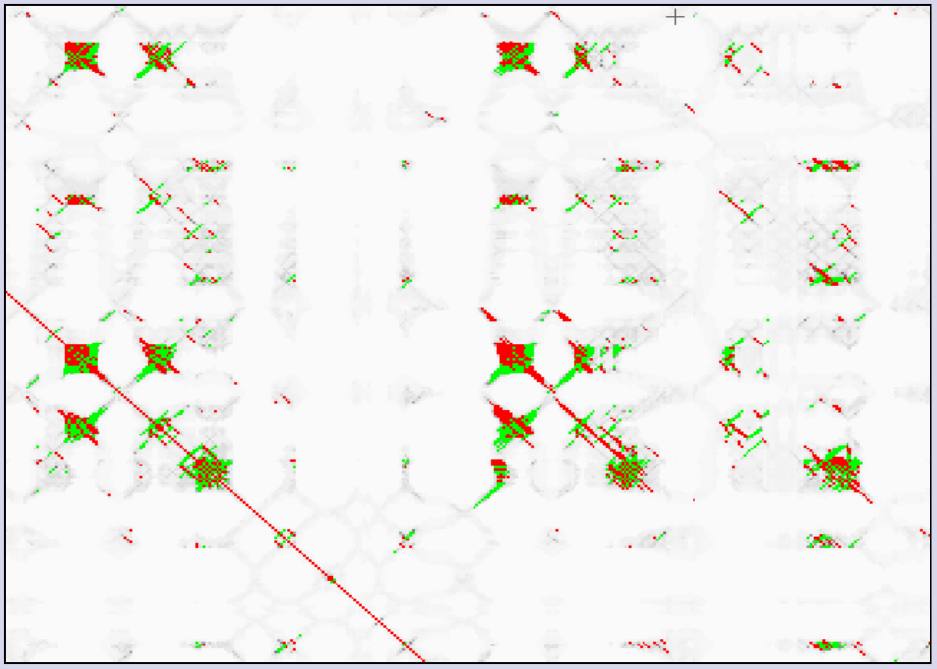


Сравнение точечной и спектральной матрицы гомологии

Точечная (OWEN)



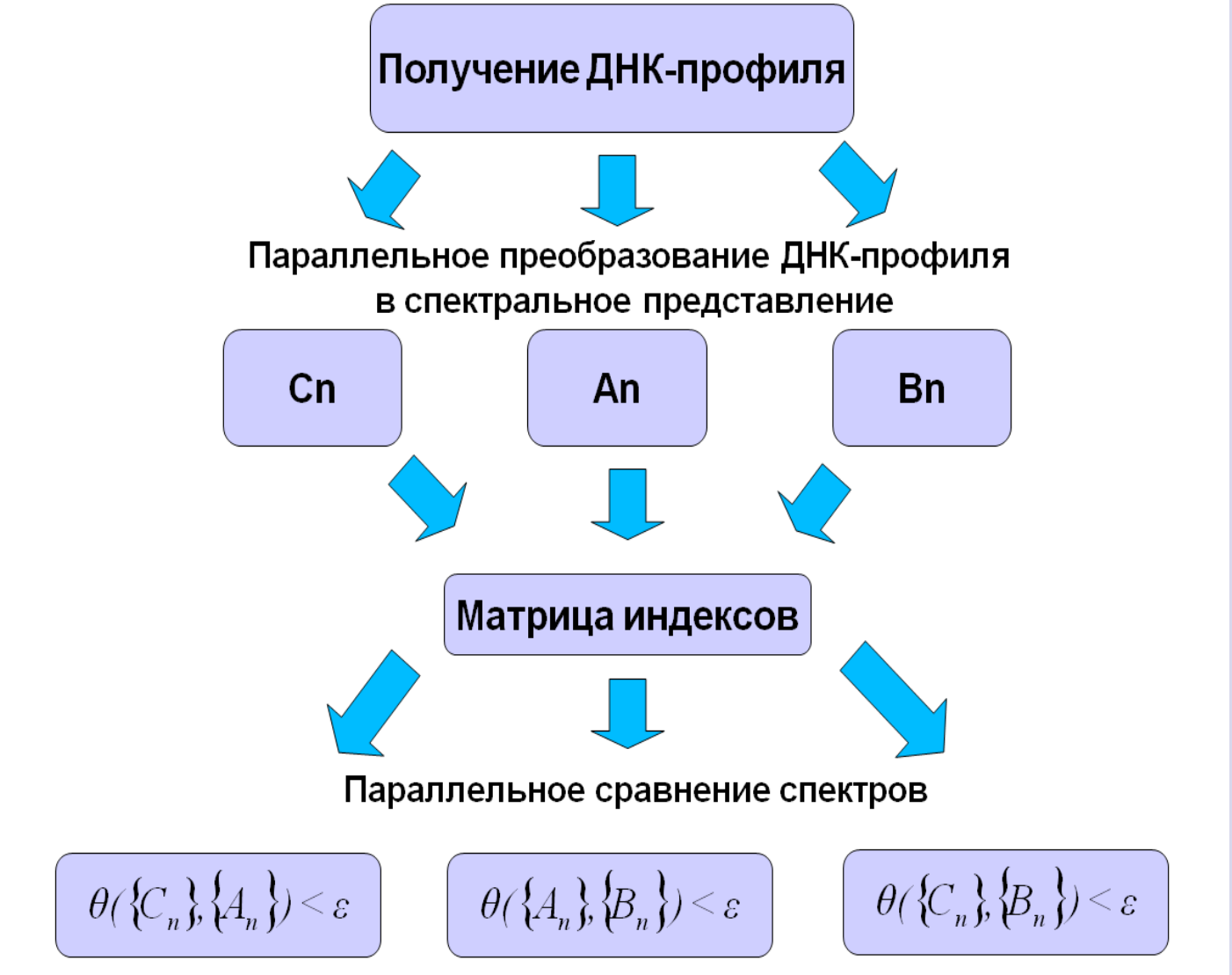
Спектральная



w2 = 5000

N = 50

Распределенная система сравнения геномов



Матрица гомологии: поиск мегасателлитных повторов