

АВТОМАТИЗИРОВАННАЯ ГЕНЕРАЦИЯ ТРЕНИРОВОЧНЫХ ЗАДАЧ И ПРОБЛЕМА АВТОМАТИЧЕСКОЙ ПРОВЕРКИ ОТВЕТОВ

Турунтаев И.С., аспирант 2 г.о.,
научный руководитель:
Данилов В.Г.

Москва 2016

ТРЕНИРОВОЧНЫЕ ЗАДАЧИ

ЗАЧЕМ АКЦЕНТИРОВАТЬ НА ЭТОМ ВНИМАНИЕ?

- Практика – это действительно важно!
- Автоматизация процесса составления и сопровождения задач увеличивает скорость и качество восприятия
- Для математических задач не подходит тестовая форма тренировки
- Это можно сделать

ТРЕНИРОВОЧНЫЕ ЗАДАЧИ

КАК ЭТО РАБОТАЕТ

- Различные задачи, посвященные общей тематике, отвечают единой структуре
- Конкретика не важна: если студент показывает хорошие результаты не ряде конкретных представителей класса задач, то можно считать, что он квалифицирован в данном классе
- Основываясь на этих двух предположениях, можно сформулировать идею шаблонизации тренировочных задач
- Традиционный пример: «*Calculate the derivative of $f(g(x))$* ».

АВТОМАТИЧЕСКАЯ ПРОВЕРКА ОТВЕТОВ

- Ответы заданы в математической нотации
- Будем рассматривать функции одной переменной
- Имеется два ответа: полученный студентом (f_{user}) и посчитанный системой (f_{real})
- Как выяснить, верен ли ответ студента? – Вычислить разность $f = f_{\text{real}} - f_{\text{user}}$ и сравнить с нулем.
- Можно ли это сделать? – Не всегда... Причина – неразрешимость Constant Problem (теорема Ричардсона, [1], [2])
- В результате: корректная проверка невозможна

АВТОМАТИЧЕСКАЯ ПРОВЕРКА ОТВЕТОВ

ЕСТЬ ЛИ РЕАЛЬНАЯ ПРОБЛЕМА?

- Может, это лишь сугубо теоретический результат и он пренебрежимо маловероятен (или вообще невозможен) на практике?
- Сумели построить пример, который поставил в тупик нашу систему

$$\frac{\sec(\operatorname{acsc}(x))}{x^2 \cdot \sqrt{1 - 1/x^2}} \cdot \tan(\operatorname{acsc}(x))$$

?

$$\frac{-1}{\left(1 - \frac{1}{x^2}\right)^{\frac{3}{2}} \cdot x^3}$$

АВТОМАТИЧЕСКАЯ ПРОВЕРКА ОТВЕТОВ

ЕСТЬ ЛИ РЕАЛЬНАЯ ПРОБЛЕМА?

- Это не полет больной фантазии, а производная
- Все зависит от того, как посмотреть на эту задачу
- Mathematica не поддалась на эту провокацию. Может, виновата наша CAS? Возможно...

$$\frac{\sec(\operatorname{acsc}(x))}{x^2 \cdot \sqrt{1 - 1/x^2}} \cdot \tan(\operatorname{acsc}(x)) = \frac{-1}{\left(1 - \frac{1}{x^2}\right)^{\frac{3}{2}} \cdot x^3}$$
$$= \frac{d}{dx} \left(\frac{1}{\sqrt{1 - \frac{1}{x^2}}} \right)$$

АВТОМАТИЧЕСКАЯ ПРОВЕРКА ОТВЕТОВ

РЕАЛЬНОСТЬ ПРОБЛЕМЫ 2

- Шаблоны => ничего не известно до конкретной генерации
- Проблемные участки могут скрываться глубоко в каких-то отдельных значениях параметров задачи
- Пользователь может ввести что угодно; уже сам по себе его ответ может испортить жизнь системе
- Ничего нельзя предугадать, невозможно даже предупредить автора шаблона о вероятных проблемах с проверкой ответа

АВТОМАТИЧЕСКАЯ ПРОВЕРКА ОТВЕТОВ

РЕАЛЬНОСТЬ ПРОБЛЕМЫ 3

- Что если в действительности с хорошим алгоритмом сравнения таких проблем на практике не возникает?
- Это бы означало, что существует некоторый класс функций, которым приводят решения всех адекватных задач и для которого не выполнена теорема Ричардсона.
- Это утверждение сомнительно
- Если это так, хотя бы обоснование его существования станет серьезным результатом

АВТОМАТИЧЕСКАЯ ПРОВЕРКА ОТВЕТОВ

КАК БЫТЬ?

- Вместо «ДА» или «НЕТ» – «НЕ ЗНАЮ». Требуется дополнительная обработка в любом случае
- Тривиальные пути разрешения:

ВАРИАНТ	ПОЧЕМУ ПЛОХО
Перегенерировать	Нет ровно никаких гарантий, что станет лучше
Всегда «НЕТ»	Совсем нечестно
Всегда «ДА»	Не очень честно, можно лучше

АВТОМАТИЧЕСКАЯ ПРОВЕРКА ОТВЕТОВ

КАК БЫТЬ?

- Рассмотрим простейший алгоритм поточенной проверки.

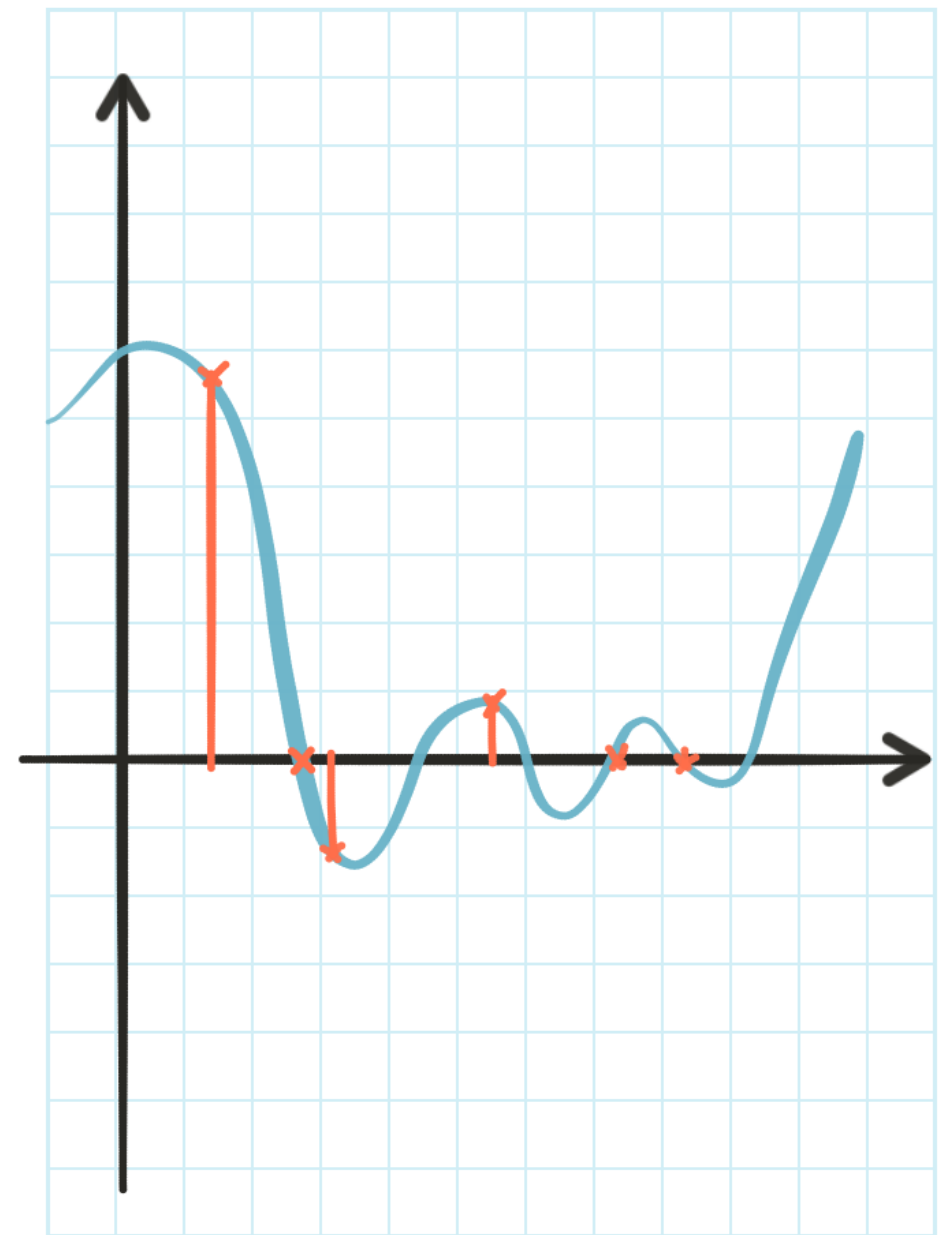
Data: f — the function to be tested,

m — a natural number,

A, B — real numbers ($A < B$)

Result: *False* or *True*

```
1 rnd = Rnd(A, B);
2 used = set();
3 f = limitExecTime(f, Δ);
4 for  $i \in \{0..m-1\}$  do
5   d = rnd.next();
6   while  $d \in \text{used}$  do
7     d = rnd.next();
8   end
9   used.add(d);
10  tmp = f(d);
11  if tmp is None then
12    i --;
13    continue;
14  end
15  if tmp != 0 then
16    return False;
17  end
18 end
19 return True;
```



АЛГОРИТМ ПОТОЧЕЧНОЙ ПРОВЕРКИ

- Рассматриваем аналитические функции на отрезке
- Точки проверки распределены равномерно на выбранном отрезке
- Не проверяем дважды одну точку
- Количество проверок ограничено
- Student-friendly
- Ошибка вполне возможна
- В этом случае можно хоть как-то оценить вероятность

АЛГОРИТМ ПОТОЧЕЧНОЙ ПРОВЕРКИ

ВЕРОЯТНОСТЬ ОШИБКИ: ИДЕАЛЬНЫЙ СЛУЧАЙ

- Алгоритм допускает ошибку лишь в том случае, когда функция не есть тождественный ноль, но все m точек «попали» в нули этой функции
- Имеем отрезок вещественной прямой.
- Функция аналитична на этом отрезке $\Rightarrow$ не может иметь бесконечное число нулей на нем
- Вероятность ошибки есть вероятность попадания в некоторое дискретное подмножество непрерывного множества, то есть ноль
- На самом деле нет.

АЛГОРИТМ ПОТОЧЕЧНОЙ ПРОВЕРКИ

ВЕРОЯТНОСТЬ ОШИБКИ: РЕАЛЬНЫЙ СЛУЧАЙ

- Речь идет о компьютерной реализации, следовательно необходимо работать в терминах машинной арифметики
- Имеем дело не с непрерывным отрезком вещественной прямой, но с сеткой чисел с плавающей точкой, аппроксимирующих вещественные числа внутри отрезка
- Таким образом, речь идет о попадании в конечное подмножество конечного множества
- Появляется ненулевая вероятность

АЛГОРИТМ ПОТОЧЕЧНОЙ ПРОВЕРКИ

ВЕРОЯТНОСТЬ ОШИБКИ

- M – число точек внутри отрезка
- m – максимальное число проверок
- Z – множество нулей функции
- k – максимальное число нулей функции внутри отрезка (оценка $|Z|$)

Вероятность выбрать первый «ноль»

$$\mathbb{P}(d_0 \in Z) = \frac{k}{M}$$

АЛГОРИТМ ПОТОЧЕЧНОЙ ПРОВЕРКИ

ВЕРОЯТНОСТЬ ОШИБКИ

- M – число точек внутри отрезка
- m – максимальное число проверок
- Z – множество нулей функции
- k – максимальное число нулей функции внутри отрезка (оценка $|Z|$)

Вероятность выбрать два нуля подряд

$$\mathbb{P}(d_0 \in Z, d_1 \in Z) = \frac{k}{M} \cdot \frac{k-1}{M-1}$$

АЛГОРИТМ ПОТОЧЕЧНОЙ ПРОВЕРКИ

ВЕРОЯТНОСТЬ ОШИБКИ

- M – число точек внутри отрезка
- m – максимальное число проверок
- Z – множество нулей функции
- k – максимальное число нулей функции внутри отрезка (оценка $|Z|$)

И так далее до m (при $m \leq k \leq M$)

$$\begin{aligned} \mathbb{P}(d_{m-1} \in Z, d_{m-1} \in Z, \dots, d_0 \in Z) &= \prod_{i=0}^{m-1} \frac{k-i}{M-i} \\ &= \frac{k!(M-m)!}{M!(k-m)!} \end{aligned}$$

АЛГОРИТМ ПОТОЧЕЧНОЙ ПРОВЕРКИ

ВЕРОЯТНОСТЬ ОШИБКИ

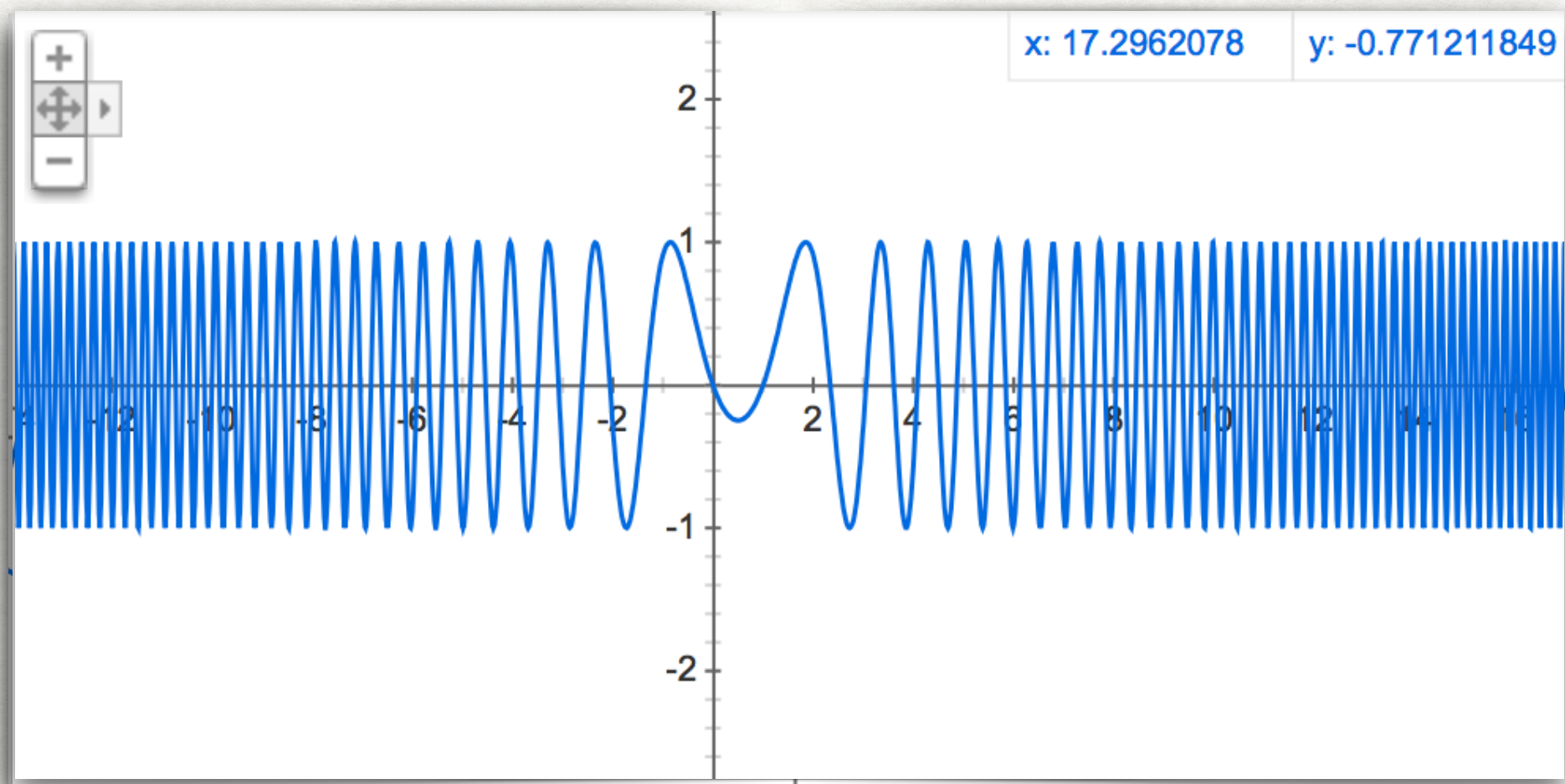
- Вероятность зависит от трех параметров: M , m и k
- m мы хотим определить из соображений минимизации вероятности ошибки при наименьшем значении самого m
- M определяется используемой численной системой
 - Это значение можно посчитать, но это чересчур трудоемко
 - Можно ограничиться следующей грубой оценкой

$$M = \left\lceil \frac{B-A}{ulp(B)} \right\rceil \leq M'$$

где M' – реальное число точек в отрезке, а $ulp(x) = b-a$, где b – ближайший справа, а a – ближайший справа к x узел сетки. [3]

АЛГОРИТМ ПОТОЧЕЧНОЙ ПРОВЕРКИ ВЕРОЯТНОСТЬ ОШИБКИ

- Рассмотрим k – верхнюю оценку на число нулей функции внутри исследуемого отрезка
- Всякое бывает...



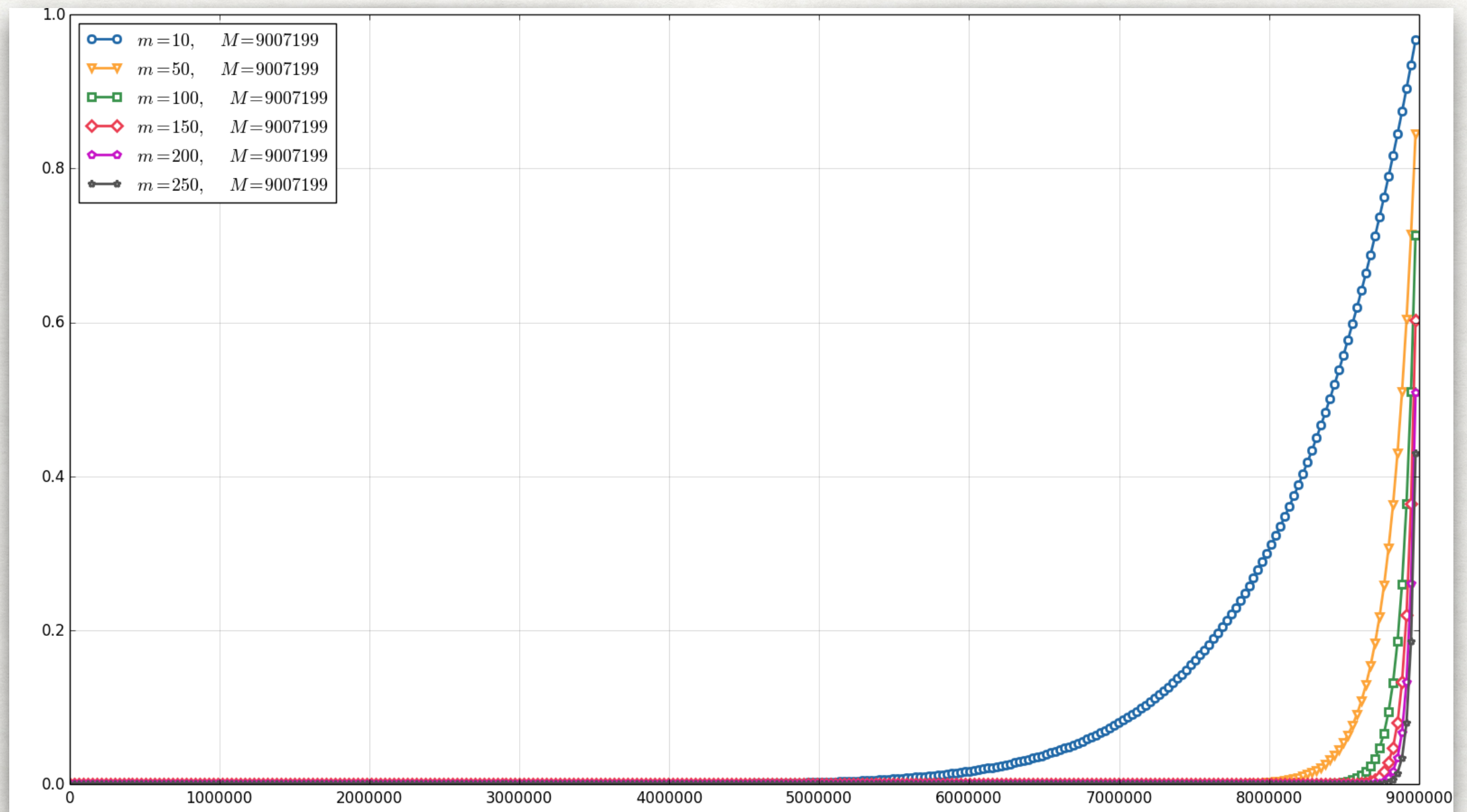
АЛГОРИТМ ПОТОЧЕЧНОЙ ПРОВЕРКИ

ВЕРОЯТНОСТЬ ОШИБКИ

- Едва ли можно достоверно оценить потенциальное число нулей
- Ограничение: мы всякий раз исследуем разность двух элементарных функций
- Плохая новость: это ничего не доказывает
- Хорошая новость: автору пока не удалось соорудить настолько противную функцию из разности
- Остается смотреть на графики

АЛГОРИТМ ПОТОЧНОЙ ПРОВЕРКИ

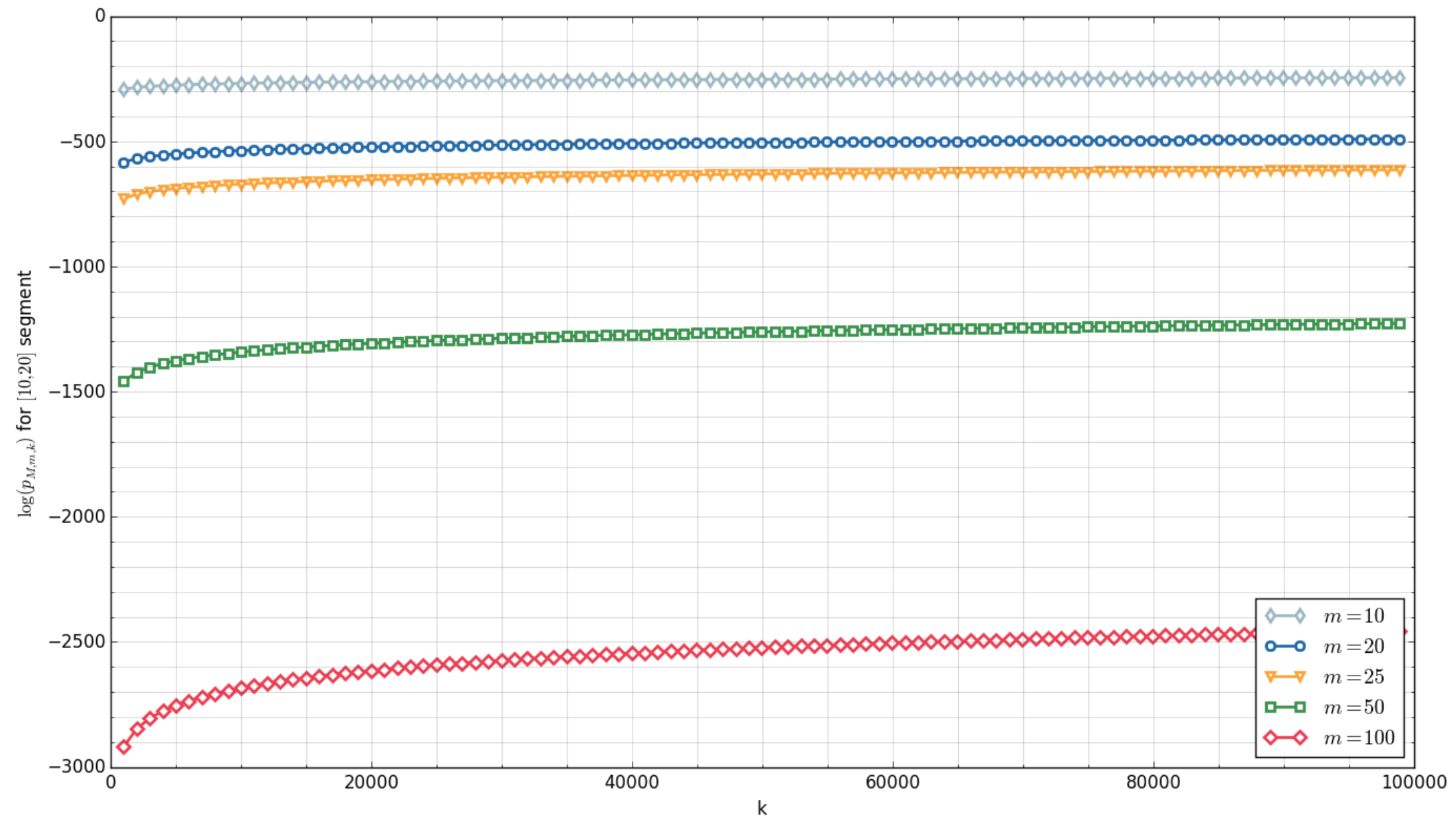
ВЕРОЯТНОСТЬ ОШИБКИ



- P в зависимости от k при различных значениях m для отрезка $[10^9, 10^9+1]$

АЛГОРИТМ ПОТОЧЕЧНОЙ ПРОВЕРКИ

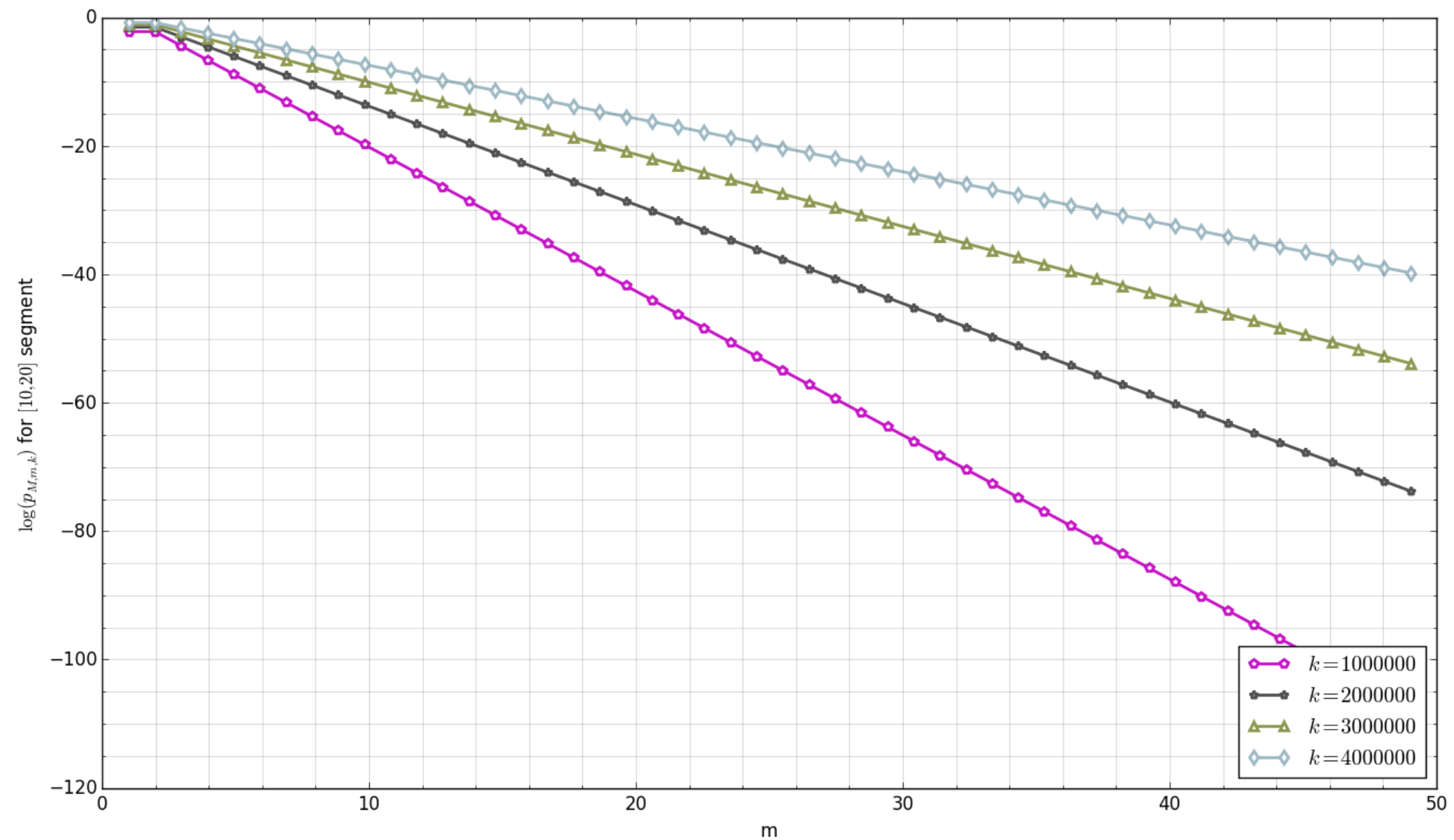
ВЕРОЯТНОСТЬ ОШИБКИ



- Логарифм вероятности по k на отрезке $[10, 20]$

АЛГОРИТМ ПОТОЧЕЧНОЙ ПРОВЕРКИ

ВЕРОЯТНОСТЬ ОШИБКИ



- Логарифм вероятности по m на отрезке $[10, 20]$

АЛГОРИТМ ПОТОЧЕЧНОЙ ПРОВЕРКИ

ВЕРОЯТНОСТЬ ОШИБКИ

- Некоторые цифры: если исследуемые функции имеют на отрезке $[10, 20]$ не более 10^6 нулей ($k = 10^6$), то достаточно проверить 10 точек ($m=10$), чтобы вероятность ошибки была не более e^{-200} .

m	k	$\log(p_{M,m,k})$
10	1000000.0	-222.281473312
10	2000000.0	-215.349979006
10	3000000.0	-211.295320425
20	1000000.0	-444.563046624
20	2000000.0	-430.700008012
20	3000000.0	-422.590674183
30	1000000.0	-666.844719939
30	2000000.0	-646.050087019
30	3000000.0	-633.886061275
40	1000000.0	-889.126493256
40	2000000.0	-861.400216026
40	3000000.0	-845.181481701
50	1000000.0	-1111.40836658
50	2000000.0	-1076.75039503
50	3000000.0	-1056.47693546

АЛГОРИТМ ПОТОЧЕЧНОЙ ПРОВЕРКИ

РЕЗУЛЬТАТ

- Строго говоря, все еще есть неопределенность
- Однако полученные численные оценки позволяют заявлять о достаточной надежности алгоритма
- Большое количество нулей исследуемой функции означает, что пользователь изобрел неправильный ответ, не зная правильного, который оказался очень близок к правильному – очень маловероятно
- В условиях неопределенности это лучший из рассмотренных вариантов для принятия окончательного решения

БЛИЖАЙШИЕ ЦЕЛИ

- В работе – автоматическое построение хода решения
- Вопрос о структуризации задач. Определение места задачи в рамках classroom. Определение на этой основе состояний студентов. Learning Spaces.
- Структуризация classrooms.
- Конструктор задач. Возможность полного отстранения от языка описания шаблонов задач.

ЛИТЕРАТУРА

1. Daniel Richardson. Some undecidable problems involving elementary functions of a real variable. *Journal of Symbolic Logic*, 33(04):514-520, 1969.
2. Miklo's Laczkovich. The removal of π from some undecidable problems involving elementary functions. *Proceedings of the American Mathematical Society*, 131(7):2235-2240, 2003.
3. David Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys (CSUR)*, 23(1):5-48, 1991.
4. John Harrison. A machine-checked theory of floating point arithmetic. In *Theorem Proving in Higher Order Logics*, pages 113-130. Springer, 1999

**СПАСИБО ЗА
ВНИМАНИЕ!**