Нижегородский государственный университет им. Н.И.Лобачевского

Факультет Вычислительной математики и кибернетики

Параллельные численные методы

Параллельные методы Монте-Карло

При поддержке компании Intel

Баркалов К.А.,
Кафедра математического обеспечения ЭВМ

Содержание

- Понятие метода Монте-Карло
  - История метода
  - Области применения
- Основы метода Монте-Карло
  - Задача численного интегрирования
  - Оценка погрешности
- Случайные и псевдослучайные числа, их генераторы
- Распараллеливание методов Монте-Карло
- Параллельная генерация псевдослучайных чисел
- Результаты экспериментов

Понятие метода статистических испытаний

- *Метод статистических испытаний (метод Монте-Карло)* – численный метод решения математических задач, при котором:
  - искомые величины представляют вероятностными характеристиками какого-либо случайного явления,
  - это явление моделируется,
  - нужные характеристики приближённо определяют путём статистической обработки «наблюдений» модели.
- Говоря о методах Монте-Карло, можно сказать ,что.
  - речь идет о численных методах (а не об аналитических)
  - решать методами Монте-Карло можно различные математические задачи (а не только задачи вероятностного происхождения, связанные со случайными величинами).

История метода Монте-Карло

- Метод статистического моделирования стал широко применяться при выполнении расчетов в рамках американского атомного проекта (Дж. фон Нейман, С. Улам).
- «Официальная дата рождения» методов Монте-Карло: 1949 год, статья под заглавием «Метод Монте-Карло».
- Происхождение названия – рулетка казино как способ генерации случайных чисел.
- Следует отметить, что:
  - теоретические основы методов Монте-Карло были известны значительно раньше;
  - фактически такие методы не раз использовались для расчетов (Бюффон, вычисление π);
  - лишь появление ЭВМ сделало метод Монте-Карло универсальным.

Области применения метода

Статистическое моделирование широко применяется для решения задач из различных областей человеческого знания.

- биология;
- химия;
- физика;
- экономика;
- ...

В некоторых областях задачи имеют очевидную вероятностную природу (задачи массового обслуживания), в некоторых – нет (численное интегрирование).

Области применения метода

Примеры задач:

- численное интегрирование,
- расчеты в системах массового обслуживания,
- расчеты качества и надежности изделий,
- расчеты прохождения нейтронов сквозь пластину,
- передача сообщений при наличии помех,
- задачи теории игр,
- задачи динамики разреженного газа,
- задачи оптимизации,
- задачи финансовой математики.

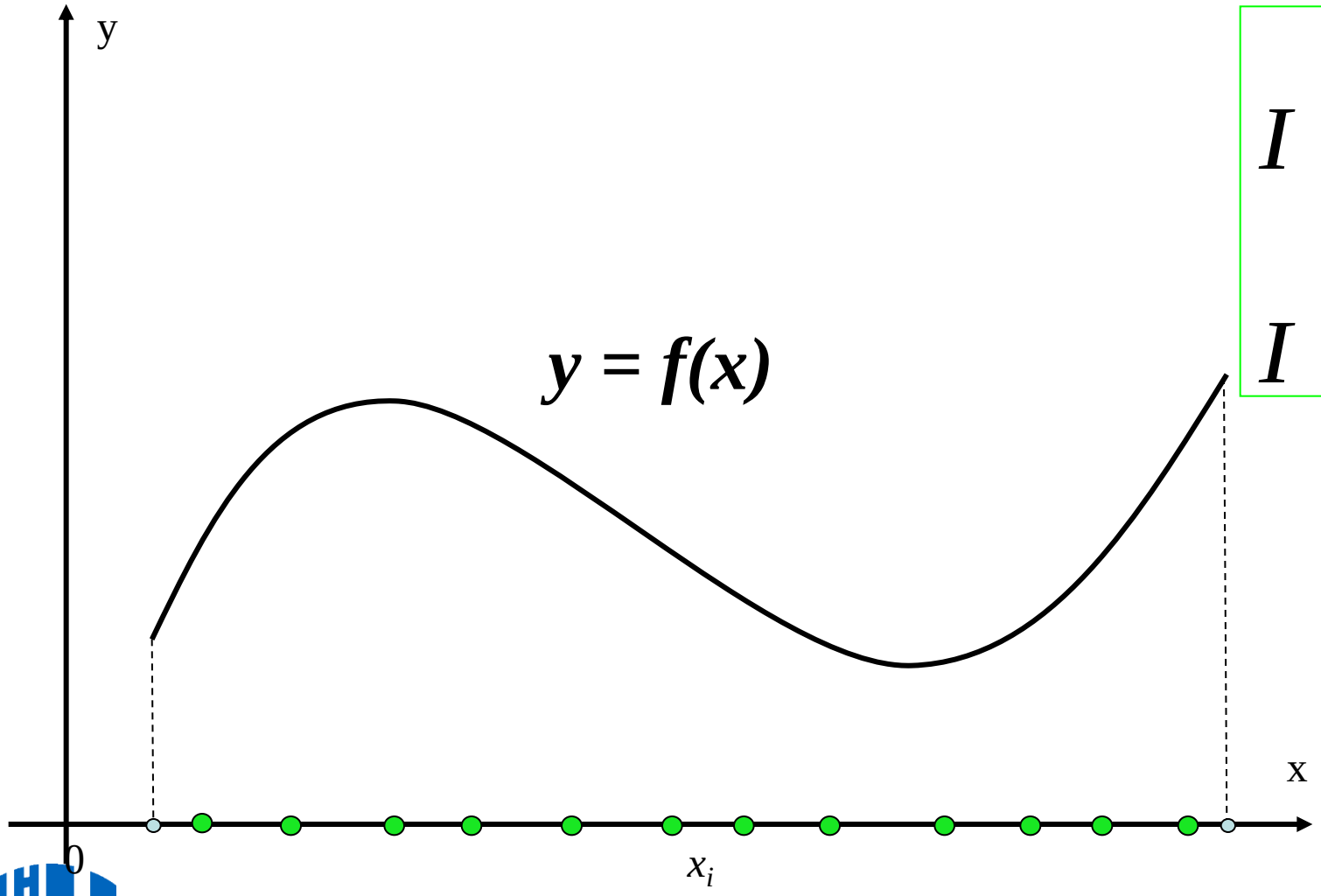
Идея метода Монте-Карло

- Сведение задачи к расчету математических ожиданий.
- Для того чтобы приближенно вычислить некоторую скалярную величину a, надо:
  - Подобрать (придумать) такую случайную величину ξ, что $M\xi = a$;
  - Пронаблюдать (вычислить) N независимых значений $\xi_1,\ldots,\xi_N$, величины ξ,
  - Оценить по выборке ее математическое ожидание как

$$M\xi \approx \frac{\xi_1 + \xi_2 + \ldots + \xi_N}{N}$$

Полученный результат можно считать *оценкой искомого значения*.

Пример – вычисление интеграла



$$I = \int_a^b f(x)dx$$

$$I = ?$$

Пример – вычисление интеграла

$$I = \int_a^b f(x)dx$$

- Рассмотрим случайную величину ξ, равномерно распределенную на $[a,b]$.

$$Mf(\xi) = \frac{1}{b-a}\int_a^b f(x)dx = \frac{1}{b-a} I \qquad I = (b-a)Mf(\xi)$$

- Оценим I по выборке размера n

$$I \approx S_n = (b-a)\sum_{i=1}^{n} f(x_i) \Big/ n$$

- Какова погрешность метода?

Оценка погрешности

- Рассмотрим случайную величину $Z=f(\xi)$
- Из теории вероятностей известно, что величина $\frac{S_n - I}{\sqrt{D(Z)/n}}$ распределена асимптотически нормально.
- При больших n погрешность вычислений будет

$$\left|S_N - I\right| \le y\sqrt{D(Z)/n}$$

с вероятностью $p_0(y)$, где $p_0(y) = 1 - \frac{\sqrt{2}}{\sqrt{\pi}} \int_y^\infty \exp\left(-\frac{t^2}{2}\right) dt$

- В частности, при y=5 $p_0(y)$=0,99999.
- Неизвестную дисперсию $D(Z)$ можно оценить как

$$D(Z) \approx \frac{1}{n}\sum_{i=1}^{n}(z_i)^2 - \left(\frac{1}{n}\sum_{i=1}^{n} z_i\right)^2$$

Оценка погрешности

- Погрешность метода Монте-Карло – $O\left(1/\sqrt{n}\right)$
  - медленно убывает;
  - зависит от дисперсии $D(Z)$;
  - не зависит от размерности задачи!
- Пусть s – размерность интеграла, ε – требуемая точность
  - Квадартурные формулы: число узлов интегрирования порядка ε^{-s}
  - Метод Монте-Карло: число случайных точек порядка ε^{-2}

Случайные числа и генераторы

- Основа методов Монте-Карло – *случайные числа*. Как их получить?
- *Генераторы случайных чисел* – устройства, которые генерируют эти числа из *непредсказуемого физического процесса*.
  - Иногда устройства основаны на *макроскопических явлениях* (непредсказуемы из-за неизвестности начальных условий)
    - Рулетка
    - Игральные кости
  - Обычно такие устройства основаны на *микроскопических явлениях*
    - «Тепловой шум» в сопротивлении или конденсаторе

Случайные числа и генераторы

- Создать физический генератор случайных чисел:
  - Дорого;
  - Сложно.
- Эксплуатировать такое устройство:
  - Дорого;
  - Сложно;
  - Медленно.
- Результаты экспериментов с использованием такого генератора – каждый раз разные.
- Вывод: обычно пользуются алгоритмическими генераторами *псевдослучайных чисел*.

Псевдослучайные числа

- Числа, получаемые по какой-либо формуле и имитирующие значения случайной величины, называются *псевдослучайными числами*. Под словом «*имитирующие*» подразумевается, что эти числа удовлетворяют ряду тестов так, как если бы они были *значениями* этой случайной величины.
- Псевдослучайные числа:
  - Получаются по некоторому алгоритму;
  - Почти независимы;
  - Удовлетворяют заданному распределению.

Генераторы псевдослучайных чисел

- *Генератор псевдослучайных чисел* (ГПСЧ, PRNG) — *алгоритм*, генерирующий последовательность чисел, элементы которой *почти независимы* друг от друга и *подчиняются заданному распределению*.
- Никакой детерминированный алгоритм не может генерировать полностью случайные числа, а только лишь аппроксимировать некоторые свойства случайных чисел.

Метод середины квадратов

- Исторически первый алгоритм получения псевдослучайных чисел предложен фон Нейманом:
  - Рассмотрим 4-значное целое число $n_1 = 9876$.
    Возведем его в квадрат. Получим 8-значное число 97 535 376.
  - Выберем четыре средние цифры из этого числа и обозначим $n_2 = 5353$.

    Затем возведем его в квадрат (28 654 609) и снова извлечем 4 средние цифры. Получим $n_3 = 6546$.
  - Далее, 42 850116, $n_4 = 8501$ и т. д.
- В качестве значений случайной величины предлагалось использовать значения 0,9876; 0,5353; 0,6546; 0,8501; 0,2670; 0,1289 и т. д.
- Недостаток – много малых значений, короткий период (повторение последовательности)

Линейный конгруэнтный генератор

- Последовательность $X_n \in [0,m)$ генерируется по формуле

$$X_{n+1} = (aX_n + c) \bmod m$$

где $a>0$, $c \geq 0$, $m>0$ – целочисленные константы.

- Свойства последовательности определяются параметрами a, c, m, а ее конкретный вид – стартовым значением X_0.
- Период последовательности X_n не превосходит m.
- Длина периода равна m тогда и только тогда, когда:
  - c и m взаимно просты;
  - $a-1$ кратно p для всех простых p – делителей m;
  - $a-1$ кратно 4, если m кратно 4.

Линейный конгруэнтный генератор

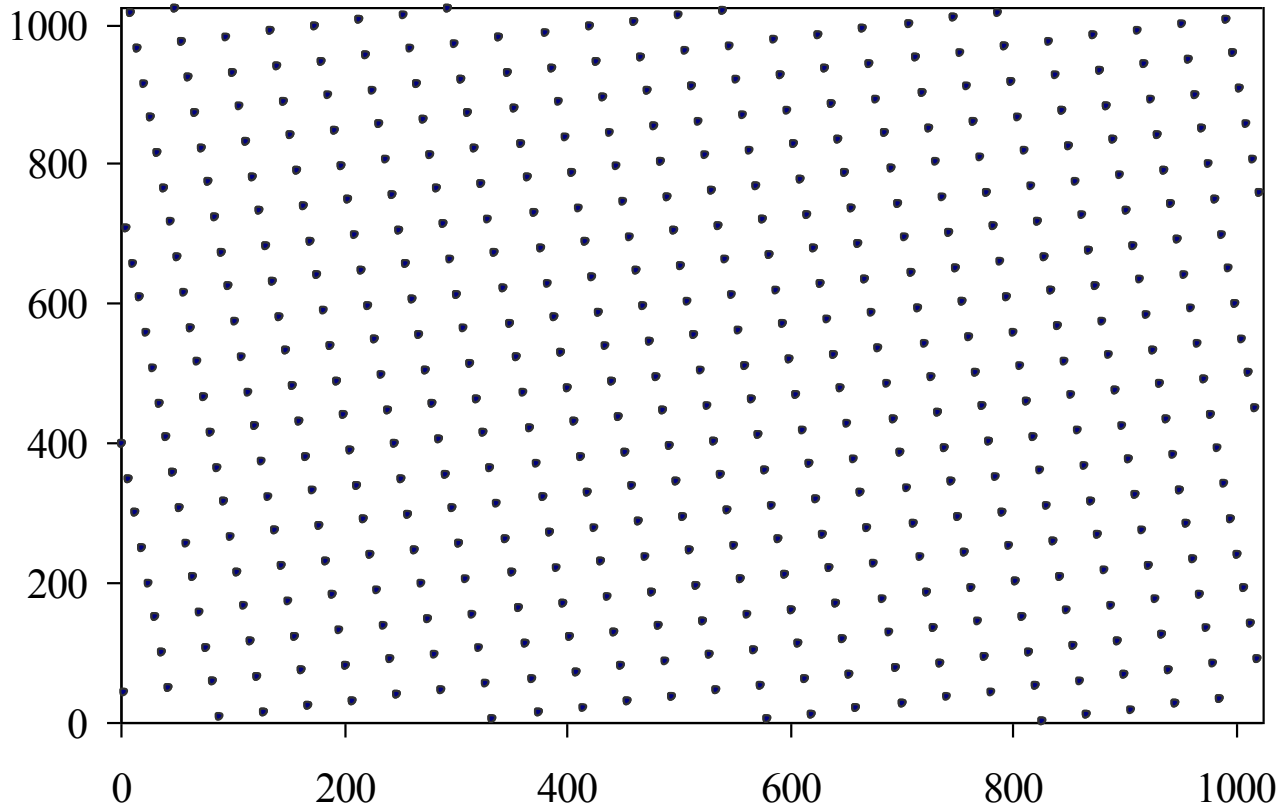
- В 32-разрядной арифметике период будет ~10^9.

  Нужно использовать 48-и или 64-х разрядную арифметику.
- Генерация вещественных чисел – по формуле

$$Y_n = X_n / (m-1)$$

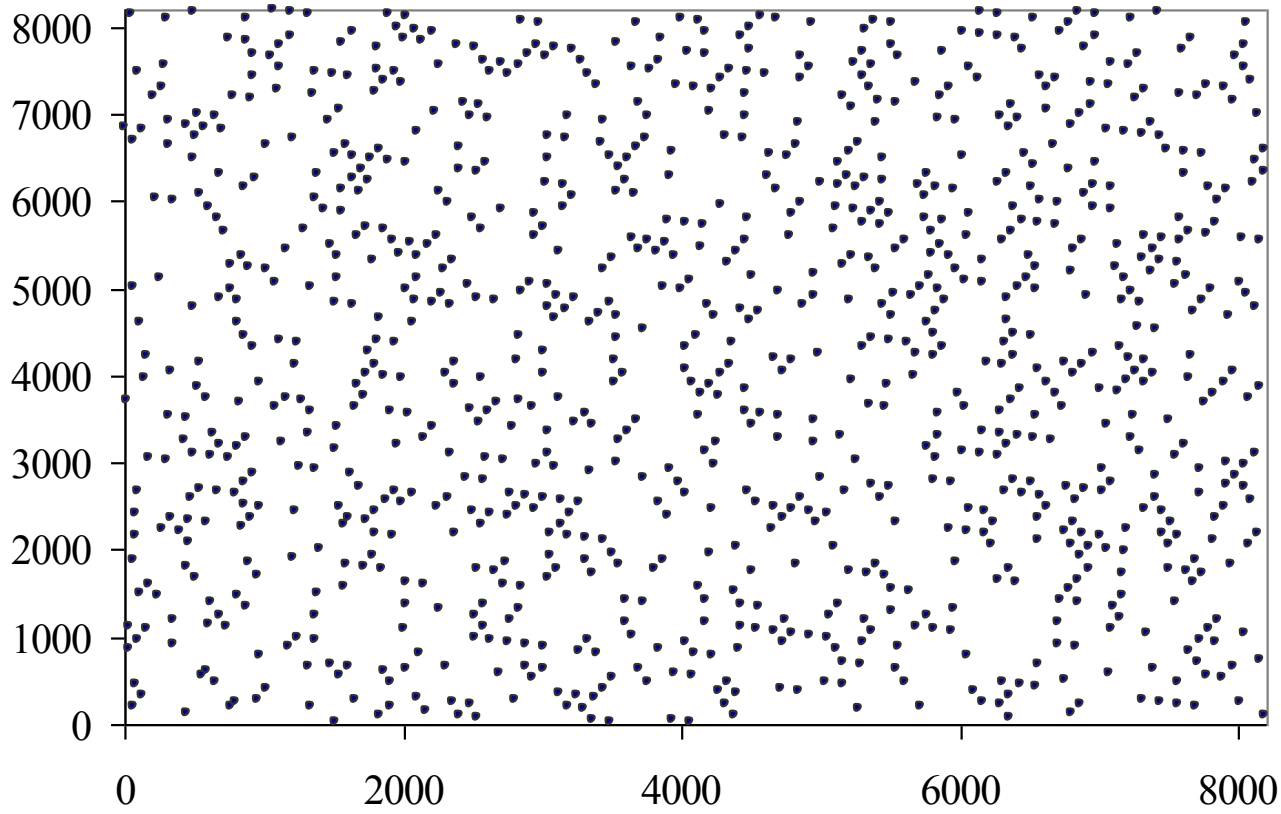
- Недостаток генератора: *d*-мерные точки (координаты – *d* подряд идущих чисел последовательности) будут расположены не более чем в $\sqrt[d]{m}$ гиперплоскостях

Линейный конгруэнтный генератор



Точки, полученные генератором $X_{n+1} = (845X_n + 2625) \bmod 1024$

Линейный конгруэнтный генератор



Точки, полученные генератором $X_{n+1} = (845X_n + 2625) \bmod 8192$

Генератор Фибоначчи

- В общем виде X_n генерируется по формуле

$$X_n = X_{n-p} * X_{n-q}$$

где $p>q>0$ – целочисленные константы (*лаги*), а символ * обозначает одну из операций: сложение, вычитание, умножение (по модулю *m*), исключающее или.

- Конкретный вид генератора последовательности $X_n \in [0,1)$

$$X_n = \begin{cases} X_{n-p} - X_{n-q}, & X_{n-p} \geq X_{n-q}; \\ X_{n-p} - X_{n-q} + 1, & X_{n-p} < X_{n-q}. \end{cases}$$

Генератор Фибоначчи

- Для работы требуется p стартовых значений $X_0, X_1, \ldots, X_{p-1}$, которые можно получить с помощью ЛК-генератора.
- Для генерации очередного числа требуется хранить p предыдущих чисел последовательности (например, в циклической очереди).
- Период последовательности оценивается как

$$T = \left(2^p - 1\right)2^{l-1}$$

где l – количество бит в мантиссе вещественного числа.

Все биты равнозначны по статистическим свойствам.

Генератор «Вихрь Мерсенна»

- В настоящее время широкое распространение получил генератор «Вихрь Мерсенна» (*Mersenne twister*).
- Описание генератора доступно в работе [5].
- Реализован во многих библиотеках (в т.ч. Intel MKL).
- Достоинства генератора:
  - Огромный период: $2^{19937}-1$
  - Равномерное распределение в 623 измерениях
  - Быстрая генерация последовательности.
- Не подходит для криптографии, т.к. некоторые тесты распознают последовательность как неслучайную.

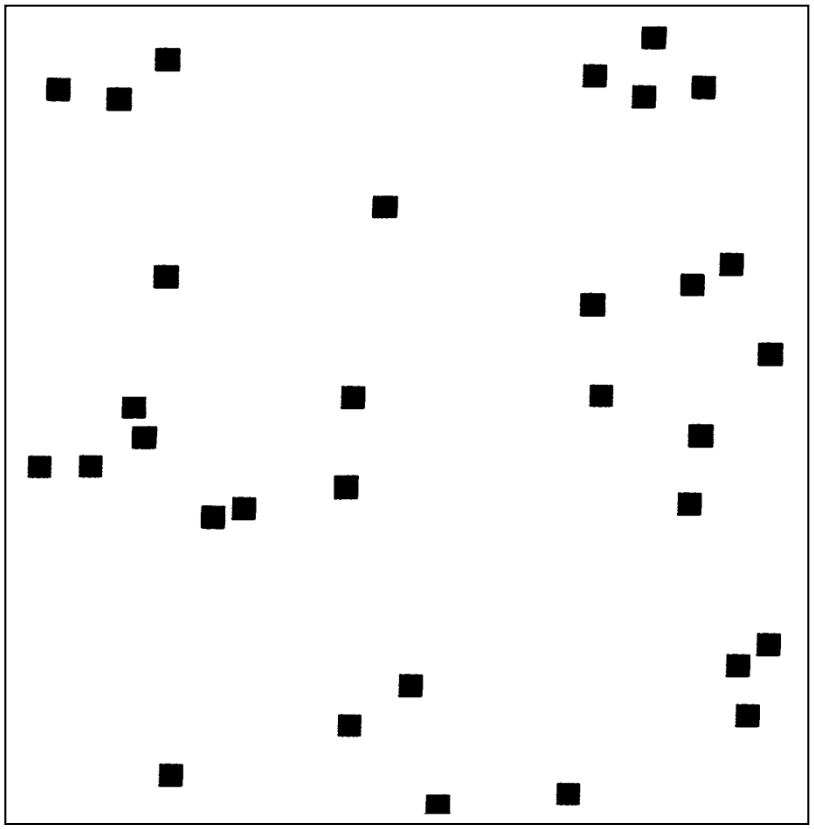
Квазислучайные числа

- При решении некоторых задач требуется, чтобы обеспечивалось «более равномерное» покрытие области значений, частично жертвуя при этом «случайностью». Числа, полученные в результате таких действий, называются *квазислучайными* (КСЧ).
- Например, в задаче численного интегрирования применение КСЧ обеспечивает существенно лучшую сходимость по сравнению с реализациями метода Монте-Карло, основанными на использовании ПСЧ.
- Генератор КСЧ может быть основан на использовании $ЛП_{\tau}$-последовательностей, детально рассмотренных в [4].

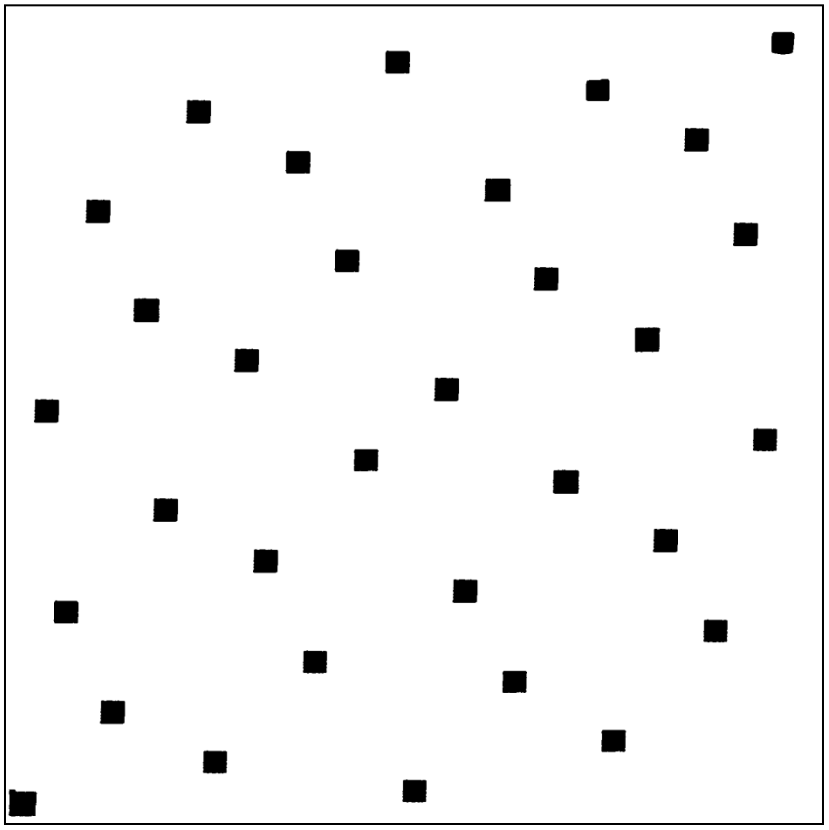
ЛП$_\tau$-последовательности

- Схема построения n-мерных точек вида $Q_i = (q_{i,1},...,q_{i,n})$ состоит в следующем.
  - Запишем номер i в двоичной системе: $i = e_m e_{m-1}...e_2 e_1$, т.е.
    $$i = e_m 2^{m-1} + e_{m-1} 2^{m-2} + ... + e_2 2^1 + e_1$$
  - Координаты точки Q_i можно получить по формуле
    $$q_{i,j} = e_1 V_j^{(1)} \oplus e_2 V_j^{(2)} \oplus ... \oplus e_m V_j^{(m)}$$
    где $\oplus$ - побитовое исключающее или, а $V_j^{(s)}$, $1 \le j \le n$, $1 \le s \le m$, координаты фиксированных *направляющих точек*
    $$V^{(1)}, V^{(2)}, ..., V^{(m)}$$
- Способ выбора направляющих точек описан в работе [4].

$ЛП_\tau$-последовательности



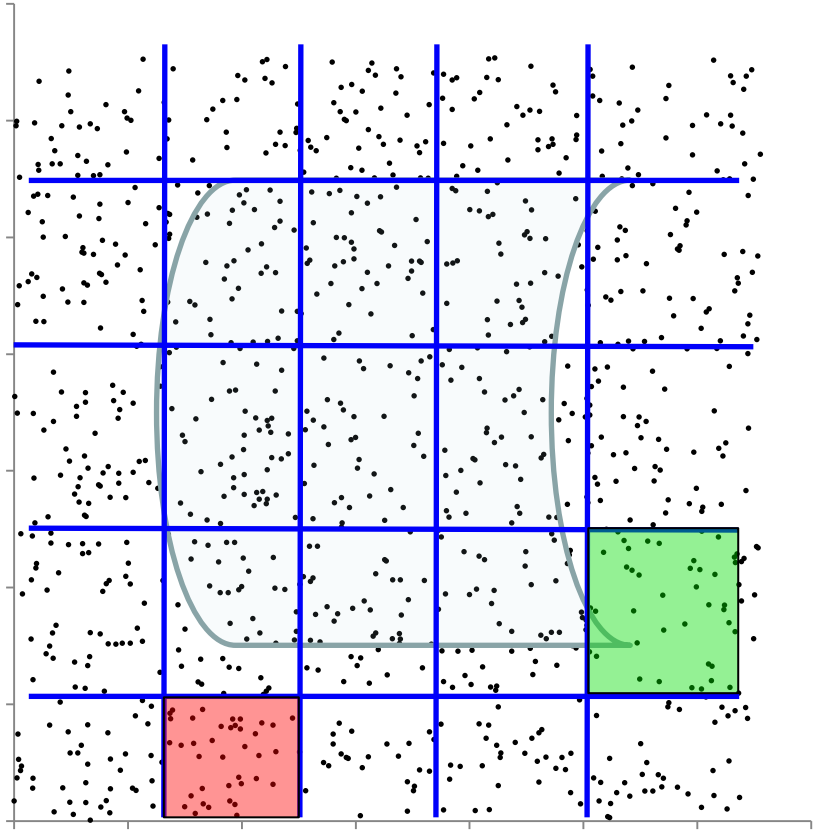
Случайные точки



Квазислучайные точки

Параллельные методы Монте-Карло

- Геометрический подход:
  - Возможен дисбаланс вычислительной нагрузки
- В каждом потоке генерируется своя последовательность
  - как ее сгенерировать?
- Нужен параллельный ГПСЧ.



Параллельные методы Монте-Карло

- Идеальные свойства парГПСЧ (+ свойства послГПСЧ):
  - Независимость
    - последовательности не должны быть коррелированы
  - Масштабируемость
    - столько последовательностей, сколько нужно
  - Локальность
    - можно создать новую последовательность без изменения всех остальных.
- Рассмотрим некоторые способы конструирования парГПСЧ из послГПСЧ.

Метод мастер-рабочий

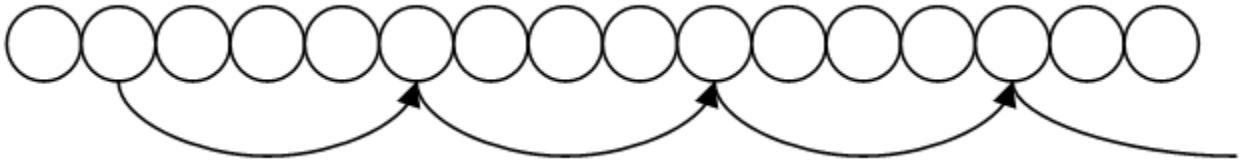
- Очевидный способ распараллеливания:
  поток-мастер, используя единственный генератор, порождает последовательность ПСЧ, и распределяет полученные числа по рабочим потокам.
- Недостатки:
  - корреляция с большим лагом в исходной последовательности может стать корреляцией с малым лагом в параллельных последовательностях;
  - метод плохо масштабируется (сложно сбалансировать скорость работы генератора в потоке-мастере и скорость обработки ПСЧ в рабочих потоках).

Метод с перешагиванием (leapfrog)

- Каждый поток работает с одинаковым генератором, порождающим последовательность X_n; при этом i-й поток обрабатывает каждое p-е число последовательности, начиная с X_i, т.е.

$$X_i, X_{i+p}, X_{i+2p}, \ldots$$

- Ниже изображены элементы последовательности, генерируемые методом leapfrog во 2-м потоке 4-х поточной программы



Использование ЛК-генератора

- Исходная формула $X_{n+1} = (aX_n + c) \bmod m$, $X_{n+k} = ?$
- Найдем (n+k)-й член последовательности

$$X_{n+2} = [aX_{n+1} + c] \bmod m = [a^2 X_n + (a+1)c] \bmod m$$

$$X_{n+3} = [aX_{n+2} + c] \bmod m = [a^3 X_n + (a^2 + a + 1)c] \bmod m$$

$$X_{n+k} = [aX_{n+k-1} + c] \bmod m = \left[a^k X_n + c \sum_{i=0}^{k-1} a^i \right] \bmod m$$

Так как $a^k - 1 = (a-1) \sum_{i=0}^{k-1} a^i$, то

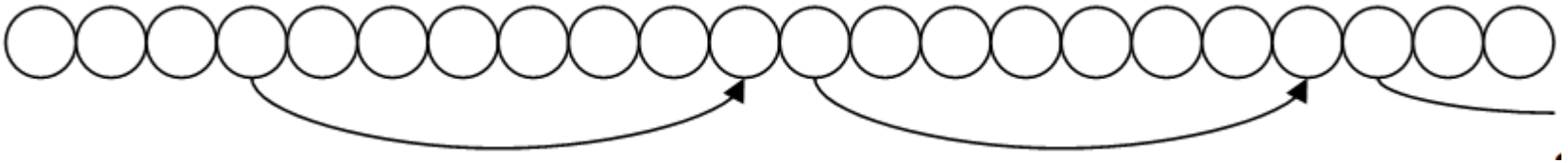
$$X_{n+k} = \left[a^k X_n + \left(\frac{a^k - 1}{a - 1} \right) c \right] \bmod m$$

Генерация точек в многомерном пространстве

- В двумерном случае: пары ПСЧ X_i, X_{i+1}, можно рассматривать как координаты точек (x_i, y_i).
- Метод с перешагиванием будет давать разные последовательности точек при разном числе потоков p.
- Чтобы точки были одинаковыми, то i-й поток должен генерировать последовательность *пар* чисел

$$X_{2i}, X_{2i+1}, X_{2i+2p}, X_{2i+2p+1} \ldots$$

- Ниже изображены ПСЧ, генерируемые методом leapfrog во 2-м потоке 4-х поточной программы

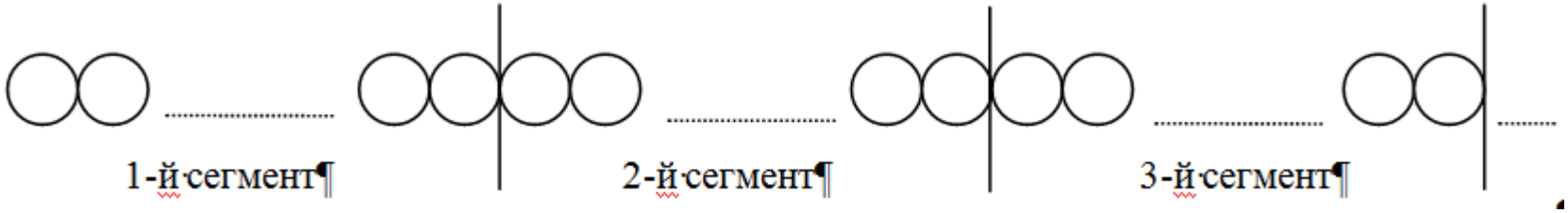


Недостатки метода leapfrog

- Сложно создать новую последовательность ПСЧ в новом потоке, т.к. каждое число заранее «закреплено» за определенным потоком;
- Элементы полученных последовательностей могут быть коррелированны для некоторых значений p. В частности, это будет происходить, если используется ЛК-генератор, m – степень двойки, и число потоков p – степень двойки.

Метод разделения последовательности

- Каждый поток работает с одинаковым генератором, порождающим последовательность X_n; при этом i-й поток обрабатывает i-й сегмент последовательности, т.е.



- Хорошо масштабируется (если период большой)
- Нужно уметь быстро вычислять n-й член, это может быть трудоемкой операцией, но ее требуется выполнить лишь один раз.

Использование ЛК-генератора

- Нужно быстро (за $O(\log_2 n)$, а не за $O(n)$) вычислять n-й член последовательности в соответствии с формулой

$$X_n = \left\lfloor a^n X_0 + \left(\frac{a^n - 1}{a - 1} \right) c \right\rfloor \bmod m$$

- Непосредственно проверяется, что

$$(a+b) \bmod m = (a \bmod m + b \bmod m) \bmod m$$
$$(ab) \bmod m = [(a \bmod m)(b \bmod m)] \bmod m$$

- Используя данные равенства, получим

$$a^n \bmod m = \left(a^{\lceil n/2 \rceil} \bmod m \cdot a^{n-\lceil n/2 \rceil} \bmod m\right) \bmod m \quad , \ldots,$$

и т.д., можно вычислить первое слагаемое за $O(\log_2 n)$ операций. Вычислим теперь второе слагаемое.

Использование ЛК-генератора

- Запишем числитель в виде $a^n - 1 = a^t\left(a^k - 1\right) + a^t - 1$ где $n=k+t$, $k = \lfloor n/2 \rfloor$, тогда

$$\left(\frac{a^n - 1}{a - 1}\right) \bmod m = \left(a^t \frac{a^k - 1}{a - 1} + \frac{a^t - 1}{a - 1}\right) \bmod m =$$

$$\left(\left(a^t\right) \bmod m \left(\frac{a^k - 1}{a - 1}\right) \bmod m + \left(\frac{a^t - 1}{a - 1}\right) \bmod m\right) \bmod m.$$

- Рекурсивно продолжая этот процесс, получим трудоемкость $O(\log_2 n)$
- Итог: вычисляем X_n за время $O(\log_2 n)$, далее вычисляем считать X_{n+i} по исходной формуле в каждом потоке.

Метод параметризации

- Идея – параллельно генерировать разные независимые последовательности ПСЧ в разных потоках:
  - Разные генераторы ПСЧ
    - Независимые последовательности;
    - Плохая масштабируемость.
  - Один генератор с разными параметрами
    - Хорошая масштабируемость;
    - Требуется тщательный выбор параметров, иначе последовательности будут коррелированны.

Метод параметризации

- ЛК-генератор: разные ПСЧ при разных константах *c*.

$$X_{n+1} = (aX_n + c) \bmod m$$

  В [6] предложено 100 констант для 100 ЛК-генераторов.
- Генератор Фибоначчи: разные ПСЧ при разных стартовых значениях $X_0, X_1, \ldots, X_{p-1}$. (нужно получить для всех потоков различные, например, методом leapfrog). Пример – библиотека SPRNG.
- Генератор Вихрь Мерсенна также допускает параметризацию. В Intel MKL – до 1024 различных генераторов Mersenne Twister.

Результаты экспериментов

- Рассмотрим задачу вычисления интеграла

$$C = e^{-rT} \int_{-\infty}^{+\infty} f(z)\varphi(z)dz\ , \qquad f(z) = \left(S_0 e^{\left(r - \frac{\sigma^2}{2}\right)T + \sigma\sqrt{T}z} - K \right)^+$$

где $\varphi(z)$ – плотность стандартного нормального распределения $N(0,1)$, а r, T, S_0, σ – числовые параметры.

- Метод Монте-Карло:

$$\hat{C} = e^{-rT} \frac{1}{N} \sum_{i=1}^{N} f(z_i) \qquad f(z_i) = \left(S_0 e^{\left(r - \frac{\sigma^2}{2}\right)T + \sigma\sqrt{T}z_i} - K \right)^+$$

где $\{z_i\}$ выбирается из $N(0,\ 1)$ (преобразование Бокса-Мюллера).

- Использовались ГПСЧ и ГКСЧ, метод разделения последовательности.

Результаты экспериментов

- Рассмотрим задачу вычисления интеграла

$$C = e^{-rT} \int_{-\infty}^{+\infty} f(z)\varphi(z)dz\ , \qquad f(z) = \left(S_0 e^{\left(r - \frac{\sigma^2}{2}\right)T + \sigma\sqrt{T}z} - K \right)^+$$

где $\varphi(z)$ – плотность стандартного нормального распределения $N(0,1)$, а r, T, S_0, σ – числовые параметры.

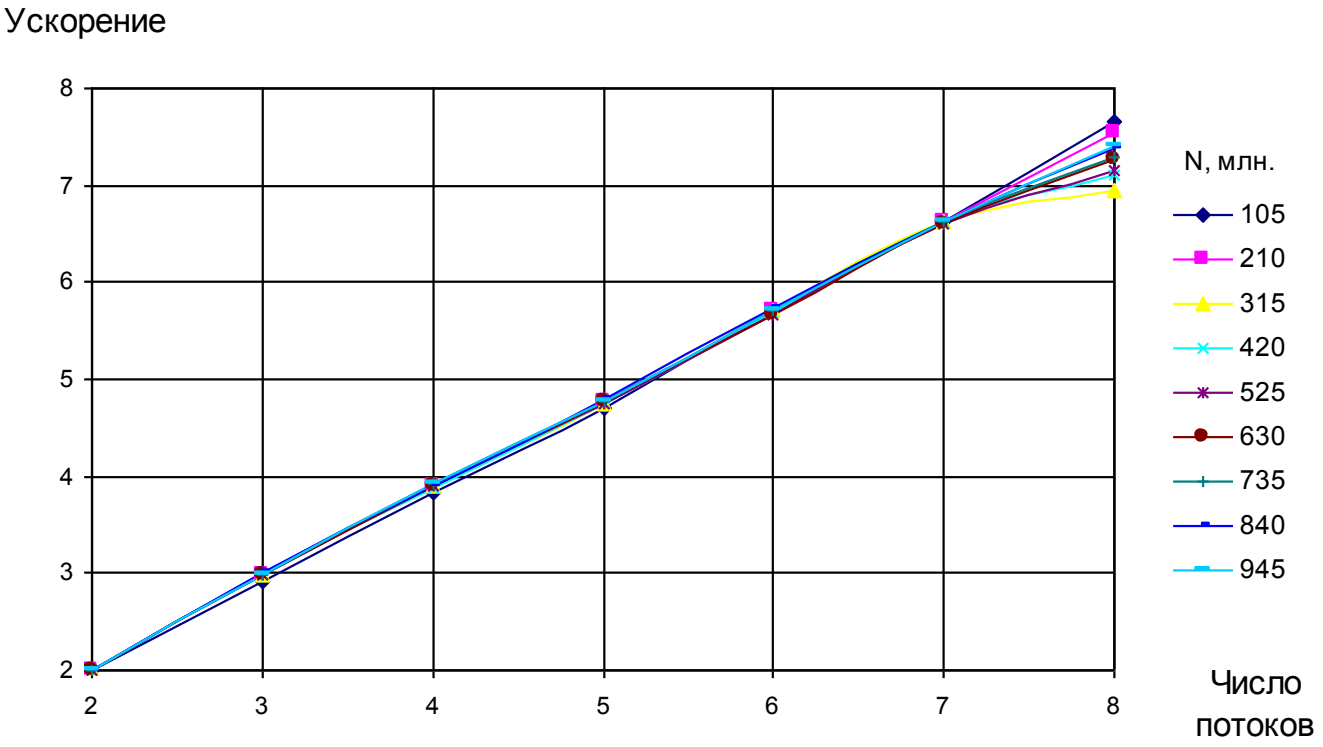
- Метод Монте-Карло:

$$\hat{C} = e^{-rT} \frac{1}{N} \sum_{i=1}^{N} f(z_i) \qquad f(z_i) = \left(S_0 e^{\left(r - \frac{\sigma^2}{2}\right)T + \sigma\sqrt{T}z_i} - K \right)^+$$

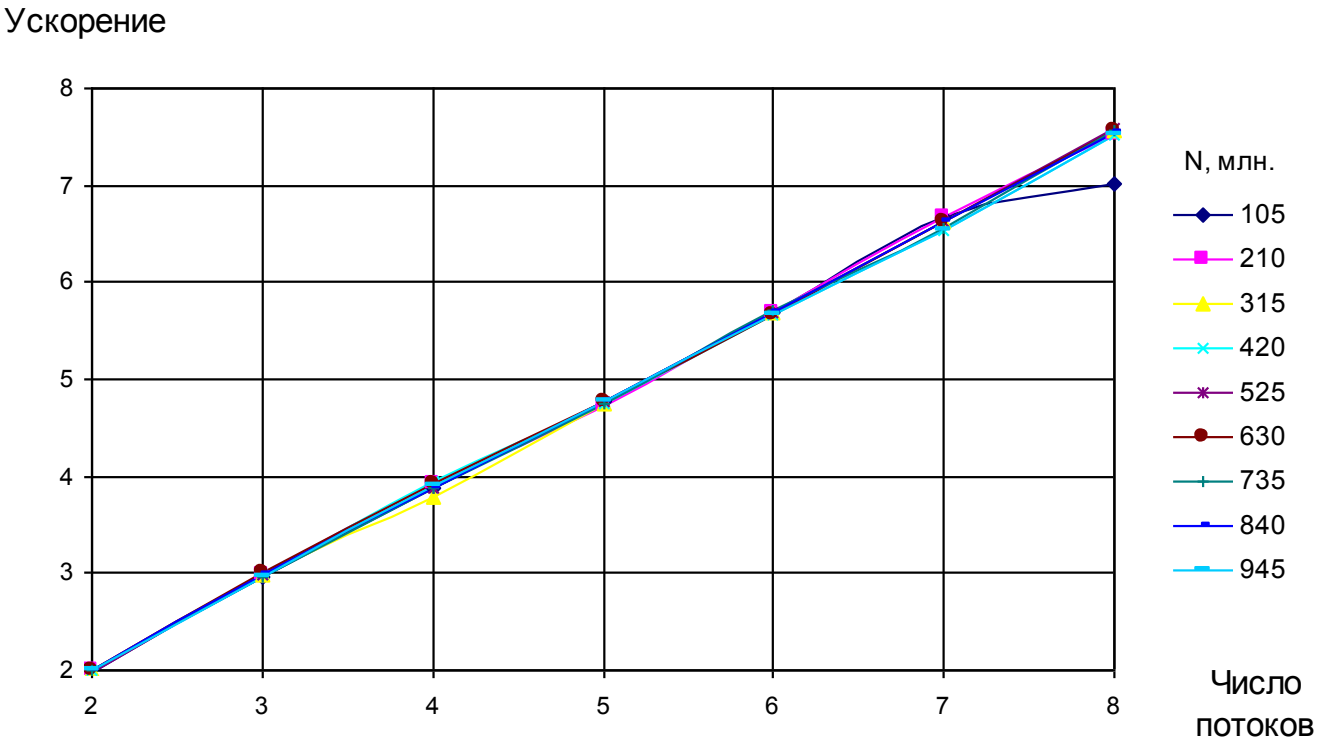
где $\{z_i\}$ выбирается из $N(0,\ 1)$ (преобразование Бокса-Мюллера).

- Использовались ГПСЧ и ГКСЧ, метод разделения последовательности.

Результаты экспериментов – ПСЧ



Результаты экспериментов – КСЧ



Заключение

- Рассмотренные вопросы:
  - История и области применения метода Монте-Карло.
  - Основы метода Монте-Карло
    - Задача численного интегрирования
    - Оценка погрешности
  - Случайные и псевдослучайные числа, их генераторы
    - Линейный конгруэнтный генератор
    - Генератор Фибоначчи
    - Вихрь Мерсенна
    - ЛП-последовательность
  - Распараллеливание методов Монте-Карло
  - Параллельная генерация псевдослучайных чисел
    - Метод мастер-рабочий
    - Метод с перешагиванием
    - Метод разделения последовательности
    - Метод параметризации
  - Результаты экспериментов

Литература

1. Соболь И.М. Численные методы Монте-Карло. – М.: Наука, 1973.
2. Соболь И.М. Точки, равномерно заполняющие многомерный куб. – М.: Знание, 1985.
3. Бахвалов Н.С., Жидков Н.П., Кобельков Г.М. Численные методы. – М.: Наука, 1987.
4. M. Quinn. Parallel programming in C with MPI and OpenMP. McGraw-Hill, 2004.
5. M. Matsumoto, T. Nishimura. «Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator». ACM Trans. on Modeling and Computer Simulations v. 8(1). 1998
6. O. Percus, M. Kalos. Random number generators for MIMD parallel processors// Journal of parallel and distributed computing, v.6, 1989.

Ресурсы сети Интернет

7. Интернет-университет суперкомпьютерных технологий. [http://www.hpcu.ru].
8. Intel Math Kernel Library Reference Manual. [http://software.intel.com/sites/products/documentation/hpc/mkl/mklman.pdf].
9. Intel Vector Statistical Library Notes. [http://software.intel.com/sites/products/documentation/hpc/mkl/vsl/vslnotes.pdf]

Авторский коллектив

- Баркалов Константин Александрович,
  к.ф.-м.н., старший преподаватель кафедры Математического обеспечения ЭВМ факультета ВМК ННГУ.
  barkalov@fup.unn.ru

- Коды учебных программ разработаны Маловой Анной и Сафоновой Яной