10.02.2025

Командная строка. Решение СЛАУ. Китайская Теорема об Остатках (КТО) и Метод Ньютона.

Филиппов Михаил Витальевич

m.filippov@g.nsu.ru

89232283872

Императивное программирование, 2024-2025

Новосибирский государственный университет

*НАСТОЯЩАЯ НАУКА

Давайте познакомимся

Филиппов Михаил Витальевич

- Окончил магистратуру ФФ НГУ
- Окончил аспирантуру ИТ СО РАН
- Являюсь м.н.с. ИТ СО РАН
- 7+ лет опыт в программировании C/C++

Новосибирский государственный университет
*НАСТОЯЩАЯ НАУКА

План лекции

Командная строка	Решение СЛАУ	Вычеты, КТО	Группы, кольца, поля (начало)
10 минут	**55 минут**	**20 минут**	**5 минут**

План лекции

Командная строка	Решение СЛАУ	Вычеты, КТО	Группы, кольца, поля (начало)
10 минут	55 минут	20 минут	5 минут

Аргументы командной строки

До появления современного графического интерфейса существовал интерфейс командной строки. Примерами могут служить DOS и Unix, к тому же терминал Linux предоставляет Unix-подобную среду командной строки.
Командная строка — это место, где вы вводите с клавиатуры информацию для запуска своей программы в среде командной строки.
Пример для программы:
$ fuss -r Ginger

Аргументы командной строки - демо

```
repeat.c
int main(int argc, char *argv[])
{

}
```

Запуск программы

repeat My name is Mikhail

argv[0] argv[1] argv[2] argv[3]

argc = 4

4 строки

```
PowerPoint/lection_x_2$ gcc main.c -o repeat
mikhail@DESKTOP-R6I9BAG:/mnt/c/Users/Mfili/Desktop/lea
PowerPoint/lection_x_2$ ./repeat My name is Mikhail
Количество аргументов, указанных в командной строке: 4
1: My
2: name
3: is
4: Mikhail
```

```c
/* repeat.c -- функция main() с аргументами */
#include <stdio.h>
int main(int argc, char *argv[])
{
    int count;
    printf("Количество аргументов, указанных в командной строке: %d\n", argc - 1);
    for (count = 1; count < argc; count++)
        printf("%d: %s\n", count, argv[count]);
    printf("\n");
    return 0;
}
```

Преобразования строк в числа

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
    int i, times;
    if (argc < 2 || (times = atoi(argv[1])) < 1)
        printf("Использование: %s положительное-число\n", argv[0]);
    else
        for (i = 0; i < times; i++)
            puts("Хорошего дня!");
    return 0;
}
// $ gcc main.c -o main
// $ ./main 3
// Хорошего дня!
// Хорошего дня!
// Хорошего дня!
```

План лекции

Командная строка	Решение СЛАУ	Вычеты, КТО	Группы, кольца, поля (начало)
10 минут	55 минут	20 минут	5 минут

СЛАУ

$$\begin{cases} a_{11} \cdot x_1 + a_{12} \cdot x_2 + \dots + a_{1n} \cdot x_n = b_1 \\ a_{21} \cdot x_1 + a_{22} \cdot x_2 + \dots + a_{2n} \cdot x_n = b_2 \\ \vdots \\ a_{m1} \cdot x_1 + a_{m2} \cdot x_2 + \dots + a_{mn} \cdot x_n = b_m \end{cases}$$

Все нужно считать для расширенной матрицы (Ab)

- Чтобы решений не было, нужно – количество линейно-независимых уравнений n>m
- Чтобы решение было единственное, нужно – количество линейно-независимых уравнений n=m
- Чтобы решение было множество, нужно – количество линейно-независимых уравнений n <m

Подробности в курсе линейной алгебры или [по ссылке].

Итак, мы можем преобразовать в матричный вид A · X = B.

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$$

Как понять, что решение 1:

Найти и исключить линейно-независиые уравнения, в итоге определитель квадратной матрицы не равен 0!

Метод Гаусса

Метод Гаусса — классический метод решения системы линейных алгебраических уравнений (СЛАУ).

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$$

Метод Гаусса решения системы линейных уравнений включает в себя 2 стадии:

- последовательное (прямое) исключение;
- обратная подстановка.

Метод Гаусса

Последовательное исключение (предположим n ≤ m, иначе множество решений)

$$\begin{cases} a_{11}\cdot x_1 + a_{12}\cdot x_2 + \cdots + a_{1n}\cdot x_n = b_1 \\ a_{21}\cdot x_1 + a_{22}\cdot x_2 + \cdots + a_{2n}\cdot x_n = b_2 \\ \vdots \\ a_{m1}\cdot x_1 + a_{m2}\cdot x_2 + \cdots + a_{mn}\cdot x_n = b_m \end{cases}$$

Делим i – ое уравнение на a_{i1}

$$\begin{cases} x_1 + \frac{a_{12}}{a_{11}}\cdot x_2 + \cdots + \frac{a_{1n}}{a_{11}}\cdot x_n = \frac{b_1}{a_{11}} \\ x_1 + \frac{a_{22}}{a_{21}}\cdot x_2 + \cdots + \frac{a_{2n}}{a_{21}}\cdot x_n = \frac{b_2}{a_{21}} \\ \vdots \\ x_1 + \frac{a_{m2}}{a_{m1}}\cdot x_2 + \cdots + \frac{a_{mn}}{a_{m1}}\cdot x_n = \frac{b_m}{a_{m1}} \end{cases}$$

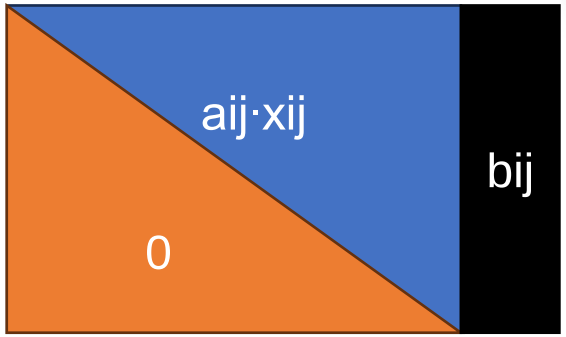
$$\begin{cases} x_1 + \frac{a_{12}}{a_{11}}\cdot x_2 + \cdots + \frac{a_{1n}}{a_{11}}\cdot x_n = \frac{b_1}{a_{11}} \\ x_1 + \left(\frac{a_{22}}{a_{21}} - \frac{a_{12}}{a_{11}}\right)\cdot x_2 + \cdots + \left(\frac{a_{2n}}{a_{21}} - \frac{a_{1n}}{a_{11}}\right)\cdot x_n = \frac{b_2}{a_{21}} - \frac{b_1}{a_{11}} \\ \vdots \\ x_1 + \left(\frac{a_{m2}}{a_{m1}} - \frac{a_{12}}{a_{11}}\right)\cdot x_2 + \cdots + \left(\frac{a_{mn}}{a_{m1}} - \frac{a_{1n}}{a_{11}}\right)\cdot x_n = \frac{b_m}{a_{m1}} - \frac{b_1}{a_{11}} \end{cases}$$

Отнимаем из каждого уравнения первое

Метод Гаусса

Последовательное исключение

$$\begin{cases} x_1 + \dfrac{a_{12}}{a_{11}} \cdot x_2 + \cdots + \dfrac{a_{1n}}{a_{11}} \cdot x_n = \dfrac{b_1}{a_{11}} \\ x_1 + \left(\dfrac{a_{22}}{a_{21}} - \dfrac{a_{12}}{a_{11}}\right) \cdot x_2 + \cdots + \left(\dfrac{a_{2n}}{a_{21}} - \dfrac{a_{1n}}{a_{11}}\right) \cdot x_n = \dfrac{b_2}{a_{21}} - \dfrac{b_1}{a_{11}} \\ \vdots \\ x_1 + \left(\dfrac{a_{m2}}{a_{m1}} - \dfrac{a_{12}}{a_{11}}\right) \cdot x_2 + \cdots + \left(\dfrac{a_{mn}}{a_{m1}} - \dfrac{a_{1n}}{a_{11}}\right) \cdot x_n = \dfrac{b_m}{a_{m1}} - \dfrac{b_1}{a_{11}} \end{cases}$$



Замена переменных

$$\begin{cases} x_1 + a'_{12} \cdot x_2 + \cdots + a'_{1n} \cdot x_n = b'_1 \\ 0 + a'_{22} \cdot x_2 + \cdots + a'_{2n} \cdot x_n = b'_2 \\ \vdots \\ 0 + a'_{m2} \cdot x_2 + \cdots + a'_{mn} \cdot x_n = b'_m \end{cases}$$

Теперь остается аналогичная система внутренний квадрат, проделываем аналогичные операции (Предположим, в итоге получилась n уравнений, m-n – оказались нулевыми – линейно зависимыми)

$$\begin{cases} x_1 + a'_{12} \cdot x_2 + \cdots + a'_{1n} \cdot x_n = b'_1 \\ 0 + x_2 + \cdots + a''_{2n} \cdot x_n = b''_2 \\ \vdots \\ 0 + 0 + \cdots + x_n = b'^{(n)}_n \\ \left.\begin{array}{c} 0 + 0 + \cdots + 0 = 0 \\ \vdots \\ 0 + 0 + \cdots + 0 = 0 \end{array}\right\} \text{m-n уравнений} \end{cases}$$

Метод Гаусса

Обратная подстановка

$$\begin{cases} x_1 + a'_{12} \cdot x_2 + \cdots + a'_{1n} \cdot x_n = b'_1 \\ 0 + x_2 + \cdots + a''_{2n} \cdot x_n = b''_2 \\ \vdots \\ 0 + 0 + \cdots + x_n = b_n^{\prime(n)} \end{cases}$$

Обратная подстановка предполагает подстановку полученного на предыдущем шаге значения переменной x_n в предыдущие уравнения. Эта процедура повторяется для всех оставшихся решений:

$$x_n = b_n^{\prime(n)}$$
$$x_{n-1} = b_{n-1}^{\prime(n-1)} - a_{(n-1)n}^{\prime(n-1)} x_n$$
$$x_{n-2} = b_{n-2}^{\prime(n-2)} - a_{(n-2)n}^{\prime(n-2)} x_n - a_{(n-2)(n-1)}^{\prime(n-2)} x_{n-1}$$
$$\ldots$$
$$x_2 = b_2'' - a_{2n}'' x_n - a''_{2\,(n-1)} x_{n-1} - \cdots - a_{23}'' x_3$$
$$x_1 = b_1' - a_{1n}' x_n - a'_{1\,(n-1)} x_{n-1} - \cdots - a_{12}' x_2$$

Метод Гаусса - пример

$$\begin{cases} 2x_1 + 4x_2 + x_3 = 36 \\ 5x_1 + 2x_2 + x_3 = 47 \\ 2x_1 + 3x_2 + 4x_3 = 37 \\ 5x_1 + x_2 + 4x_3 = 48 \end{cases}$$

$$\begin{cases} x_1 + 2x_2 + \frac{1}{2}x_3 = 18 \\ x_1 + \frac{2}{5}x_2 + \frac{1}{5}x_3 = \frac{47}{5} \\ x_1 + \frac{3}{2}x_2 + 2x_3 = \frac{37}{2} \\ x_1 + \frac{1}{5}x_2 + \frac{4}{5}x_3 = \frac{48}{5} \end{cases}$$

$$\begin{cases} x_1 + 2x_2 + \frac{1}{2}x_3 = 18 \\ -\frac{8}{5}x_2 - \frac{3}{10}x_3 = -\frac{43}{5} \\ -\frac{1}{2}x_2 + \frac{3}{2}x_3 = \frac{1}{2} \\ -\frac{9}{5}x_2 + \frac{3}{10}x_3 = -\frac{42}{5} \end{cases}$$

$$\begin{cases} x_1 + 2x_2 + \frac{1}{2}x_3 = 18 \\ x_2 + \frac{3}{16}x_3 = \frac{43}{8} \\ x_2 - 3x_3 = -1 \\ x_2 - \frac{1}{6}x_3 = \frac{14}{3} \end{cases}$$

$$\begin{cases} x_1 + 2x_2 + \frac{1}{2}x_3 = 18 \\ x_2 + \frac{3}{16}x_3 = \frac{43}{8} \\ -\frac{51}{16}x_3 = -\frac{51}{8} \\ -\frac{17}{48}x_3 = -\frac{17}{24} \end{cases}$$

$$\begin{cases} x_1 + 2x_2 + \frac{1}{2}x_3 = 18 \\ x_2 + \frac{3}{16}x_3 = \frac{43}{8} \\ x_3 = 2 \\ x_3 = 2 \end{cases}$$

$$\begin{cases} x_1 + 2x_2 + \frac{1}{2}x_3 = 18 \\ x_2 + \frac{1}{4}x_3 = \frac{43}{8} \\ x_3 = 2 \\ 0 = 0 \end{cases}$$

$$x_3 = 2$$

$$x_2 = -\frac{3}{16}x_3 + \frac{43}{8} = \frac{43}{8} - \frac{3}{8} = 5$$

$$x_1 = 18 - 2x_2 - \frac{1}{2}x_3 = 18 - 10 - 1 = 7$$

Решение систем линейных уравнений

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$$

$$A \cdot X = B$$

Если матрица A невырождена, имеется обратная к ней матрица A^{-1}, и

$$x = A^{-1}b$$

является вектором решения. Можно доказать, что x является единственным решением уравнения, следующим образом. Если имеется два решения, x и x', то Ax = Ax' = b и, обозначая тождественную матрицу как I,

$$x = Ix = (A^{-1}A)x = A^{-1}(Ax) = A^{-1}(Ax') = (A^{-1}A)x' = x'.$$

Рассмотрим решения системы линейных уравнений Ax = b из n уравнений от n неизвестных. Мы можем вычислить A^{-1}, а затем, воспользовавшись уравнением $x = A^{-1}b$, умножить b на A^{-1} и получить $x = A^{-1}b$. На практике этого подхода избегают из-за его численной неустойчивости. К счастью, другой подход - LUP-разложение - численно устойчив и обладает тем преимуществом, что на практике оказывается более быстрым.

Обзор LUP-разложения

Идея, лежащая в основе LUP-разложения, состоит в поиске трех матриц, L, U и P, размером n × n, таких, что

$$PA = LU$$

где

- L - единичная нижнетреугольная матрица;
- U - верхнетреугольная матрица;
- P - матрица перестановки.

Матрицы L, U и P, удовлетворяющие уравнению, называются LUP-разложением (LUP decomposition) матрицы A.

Преимущество вычисления LUP-разложения матрицы A основано на том, что система линейных уравнений решается гораздо легче, если ее матрица треугольна, что и выполняется в случае матриц L и U. Найдя LUP-разложение матрицы A, мы можем решить уравнение Ax = b, путем решения треугольной системы линейных уравнений. Умножая обе части уравнения Ax = b на P, мы получим эквивалентное уравнение PAx = Pb. Используя разложение $PA = LU$, получаем

Обзор LUP-разложения

$$LUx = Pb.$$

Теперь можно решить полученное уравнение, решив две треугольные системы линейных уравнений. Обозначим $y = Ux$, где x - решение исходной системы линейных уравнений. Сначала решим нижнетреугольную систему линейных уравнений

$$Ly = Pb$$

найдя неизвестный вектор y с помощью метода, который называется прямой подстановкой. После этого, имея вектор y, решим верхнетреугольную систему линейных уравнений

$$Ux = y,$$

найдя x с помощью обратной подстановки. Поскольку матрица перестановки обращаема, умножение обеих частей уравнения на P^{-1} дает $P^{-1}PA = P^{-1}LU$, так что

$$A = P^{-1}LU.$$

Следовательно, вектор является искомым решением системы линейных уравнений $Ax = b$:

$$Ax = P^{-1}LUx = P^{-1}Ly = P^{-1}Pb = b.$$

Прямая и обратная подстановки

Прямая подстановка (forward substitution) позволяет решить нижнетреугольную систему линейных уравнений для данных L, P и за время $\Theta(n^2)$. Для удобства мы представим перестановку P в компактной форме с помощью массива π[1…n]. Элемент π[i] при i = 1, 2, …, n указывает, что $P_{i,\pi[i]} = 1$ и $P_{ij} = 0$ для $j \neq \pi[i]$. Таким образом, в матрице PA на пересечении i-й строки и j-го столбца находится элемент $a_{\pi[i],j}$, а i-м элементом Pb является $b_{\pi[i]}$. Поскольку L является единичной нижнетреугольной матрицей, уравнение можно переписать следующим образом:

$$
\begin{aligned}
y_1 \qquad\qquad\qquad\qquad\qquad\qquad &= b_{\pi[1]},\\
l_{21}y_1 + \quad y_2 \qquad\qquad\qquad\qquad &= b_{\pi[2]},\\
l_{31}y_1 + l_{32}y_2 + \quad y_3 \qquad\qquad &= b_{\pi[3]},\\
\\
l_{n1}y_1 + l_{n2}y_2 + l_{n3}y_3 + \cdots + y_n &= b_{\pi[n]},
\end{aligned}
$$

Из первого уравнения получаем $y_1 = b_{\pi[1]}$. Зная значение y_1, его можно подставить во второе уравнение

$$y_2 = b_{\pi[2]} - l_{21}y_1$$

Теперь можно подставить в третье уравнение два найденных значения, y_1 и y_2, и получить

$$y_3 = b_{\pi[3]} - l_{31}y_1 - l_{32}y_2$$

В общем случае для поиска мы подставляем найденные значения y_{i-1} в i-е уравнение и находим

$$y_i = b_{\pi[i]} - \sum_{j=1}^{i-1} l_{ij}y_j$$

Прямая и обратная подстановки

После того как мы нашли y, найти x можно, воспользовавшись обратной подстановкой (back substitution), которая аналогична прямой. В этом случае мы решаем первым n-е уравнение и работаем в обратном направлении, к первому уравнению. Как и прямая подстановка, этот процесс требует времени Ө(n^2). Поскольку является верхнетреугольной матрицей, систему линейных уравнений можно переписать как

$$u_{11}x_1 + u_{12}x_2 + \cdots + u_{1,n-2}x_{n-2} + u_{1,n-1}x_{n-1} + u_{1,n}x_n = y_1,$$

$$u_{22}x_2 + \cdots + u_{2,n-2}x_{n-2} + u_{2,n-1}x_{n-1} + u_{2,n}x_n = y_2,$$

$$u_{n-2,n-2}x_{n-2} + u_{n-2,n-1}x_{n-1} + u_{n-2,n}x_n = y_{\mathrm{n}-2},$$

$$u_{n-1,n-1}\, x_{n-1} + u_{n-1,n}x_n = y_{\mathrm{n}-1},$$

$$u_{n,n}x_n = y_{\mathrm{n}}.$$

Таким образом, можно последовательно найти x_n, x_{n-1},…, x_1 следующим образом:

$$x_n = y_{\mathrm{n}}/u_{n,n}\,,$$

$$x_{n-1} = \left(y_{\mathrm{n}-1} - u_{n-1,n}x_n\right)/u_{n-1,n-1}\,,$$

$$x_{n-2} = \left(y_{\mathrm{n}-2} - u_{n-2,n-1}x_{n-1} - u_{n-2,n}x_n\right)/u_{n-2,n-2}\,,$$

или, в общем случае,

$$x_i = (y_{\mathrm{i}} - \sum_{j=i+1}^{n} u_{ij}x_j)/u_{ii}$$

Прямая и обратная подстановки

Процедура LUP-SOLVE для заданных P, L, U и b находит x путем комбинирования прямой и обратной подстановок. В псевдокоде предполагается, что размерность n хранится в атрибуте L.rows и что матрица перестановки P представлена массивом π.

LUP-Solve(L,U,r, b)
1 n = L.rows
2 Пусть x и y - вновь созданные векторы длиной n
3 for i = 1 to n
4 $y_i = b_{\pi[i]} - \sum_{j=1}^{i-1} l_{ij} y_j$
5 for i = n downto l
6 $x_i = (y_i - \sum_{j=i+1}^{n} u_{ij} x_j)/u_{ii}$
7 return x

Процедура LUP-SOLVE находит y в строках 3 и 4 с помощью прямой подстановки, а затем вычисляет x с помощью обратной подстановки в строках 5 и 6.

Наличие внутри каждого цикла for неявного цикла суммирования приводит ко времени работы данной процедуры, равному $\Theta(n^2)$.

Прямая и обратная подстановки - пример

$$\begin{pmatrix} 1 & 2 & 0 \\ 3 & 4 & 4 \\ 5 & 6 & 3 \end{pmatrix} x = \begin{pmatrix} 3 \\ 7 \\ 8 \end{pmatrix},$$

где

$$A = \begin{pmatrix} 1 & 2 & 0 \\ 3 & 4 & 4 \\ 5 & 6 & 3 \end{pmatrix},$$

$$b = \begin{pmatrix} 3 \\ 7 \\ 8 \end{pmatrix},$$

и которую необходимо решить, найдя неизвестный вектор x. LUP-разложение матрицы A имеет вид

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 0.2 & 1 & 0 \\ 0.6 & 0.5 & 1 \end{pmatrix},$$

$$U = \begin{pmatrix} 5 & 6 & 3 \\ 0 & 0.8 & -0.6 \\ 0 & 0 & 2.5 \end{pmatrix},$$

$$P = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix},$$

Используя прямую подстановку, находим y из Ly = Pb:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0.2 & 1 & 0 \\ 0.6 & 0.5 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 8 \\ 3 \\ 7 \end{pmatrix},$$

получив

$$y = \begin{pmatrix} 8 \\ 1.4 \\ 1.5 \end{pmatrix},$$

путем вычисления сначала y_1, затем – y_2 и наконец – y_3. Затем с помощью обратной подстановки находим x из Ux = y:

$$\begin{pmatrix} 5 & 6 & 3 \\ 0 & 0.8 & -0.6 \\ 0 & 0 & 2.5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 8 \\ 1.4 \\ 1.5 \end{pmatrix},$$

тем самым получив искомое решение исходной системы линейных уравнений

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 8 \\ 1.4 \\ 1.5 \end{pmatrix}$$

путем вычисления сначала x_3, затем - x_2 и наконец - x_1.

Вычисление LU-разложения

Мы показали, что если можно вычислить LUP-разложение невырожденной матрицы A, то для решения системы линейных уравнений Ax = b можно воспользоваться простыми прямой и обратной подстановками. Осталось показать, как эффективно найти LUP-разложение матрицы A. Начнем со случая невырожденной матрицы A размером n x n и отсутствия матрицы P (или, что эквивалентно, $P = I_n$). В этом случае необходимо найти разложение A = LU. Матрицы L и U называются LU-разложением (LU decomposition) матрицы A.

Итак, мы хотим построить LU-разложение невырожденной матрицы A размером n × n. Если n = 1, задача решена, поскольку мы можем выбрать и U = A. При n > 1 разобьем A на четыре части:

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} & w^T \\ v & A' \end{pmatrix}.$$

где $v = (v_2, v_3, \ldots, u_n) = (a_{21}, a_{22}, \ldots, a_{n1})$ представляет собой вектор-столбец длиной (n - 1), $w^T = (w_2, w_3, \ldots, w_n)^T = (a_{12}, a_{13}, \ldots, a_{in})^T$ является вектором-строкой длиной (n - 1), а A' - матрицей размером (n - 1) × (n - 1).

Вычисление LU-разложения

Используя матричную алгебру (проверить полученный результат можно с помощью умножения), разложим матрицу A следующим образом:

$$A = \begin{pmatrix} a_{11} & w^T \\ v & A' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ v/a_{11} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{11} & w^T \\ 0 & A' - vw^T/a_{11} \end{pmatrix}$$

Полученная в результате матрица размером (n - 1) × (n - 1)

$$A' - \frac{vw^T}{a_{11}}$$

называется ***дополнением Шура (Schur complement)*** матрицы A по отношению к элементу a_{11}. Мы утверждаем, что если матрица A невырождена, то невырождено и дополнение Шура. Почему? **ДЗ.** Поскольку дополнение Шура невырождено, можно рекурсивно найти его LU-разложение. Запишем

$$A' - vw^T/a_{11} = L'U'$$

где L' - единичная нижнетреугольная матрица, а U' – верхнетреугольная матрица. Тогда, прибегнув к матричной алгебре, получим

$$A = \begin{pmatrix} 1 & 0 \\ v/a_{11} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{11} & w^T \\ 0 & A' - vw^T/a_{11} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ v/a_{11} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{11} & w^T \\ 0 & L'U' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ v/a_{11} & L' \end{pmatrix} \begin{pmatrix} a_{11} & w^T \\ 0 & U' \end{pmatrix} = LU$$

что дает искомое LU-разложение. Причина, по которой в LUP-разложение включается матрица перестановок P, состоит в том, чтобы избежать деления на нулевые элементы. Использование матрицы перестановки для того, чтобы избежать деления на нуль (или на малые величины, что вносит вклад в численную неустойчивость), называется выбором ведущего элемента (pivoting).

Вычисление LU-разложения

LU-Decomposition(A)

n = A.rows

L и U являются новыми матрицами размером n × n

Инициализируем U нулями ниже диагонали

Инициализируем L единицами на диагонали и нулями выше нее

for k = 1 to n

    $u_{kk} = a_{kk}$

  for i = k + 1 to n

    $l_{ik} = a_{ik}/a_{kk}$ // a_{ik} хранит v_i

    $u_{ki} = a_{ki}$ // a_{ki} хранит w_i

  for i = k + 1 to n

    for j = k + 1 to n

      $a_{ij} = a_{ij} - l_{ik}l_{kj}$

return L и U

Вычисление LU-разложения

Работа процедуры LU-Decomposition. Здесь проиллюстрирована стандартная оптимизация процедуры, при которой мы храним значащие элементы L и U без привлечения дополнительной памяти, непосредственно в матрице A. Иначе говоря, мы можем установить соответствие между каждым элементом a_{ij} и либо l_{ij} (если $i > j$), либо u_{tj} (если $i \le j$) и обновлять матрицу A так, чтобы по окончании работы процедуры в ней хранились матрицы L и U. Чтобы получить псевдокод этой оптимизации из приведенного выше, необходимо просто заменить каждое обращение к l и u на обращение к a; можно легко убедиться в том, что такое преобразование сохраняет корректность алгоритма.

2	3	1	5
6	13	5	19
2	19	10	23
4	10	11	31

2	3	1	5
3	4	2	4
1	16	9	18
2	4	9	21

2	3	1	5
6	**4**	2	4
2	4	1	2
4	1	7	17

2	3	1	5
6	4	2	4
2	4	**1**	2
4	1	7	3

$$\underbrace{\begin{pmatrix} 2 & 3 & 1 & 5 \\ 6 & 13 & 5 & 19 \\ 2 & 19 & 10 & 23 \\ 4 & 10 & 11 & 31 \end{pmatrix}}_{A} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 2 & 1 & 7 & 1 \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} 2 & 3 & 1 & 5 \\ 0 & 4 & 2 & 4 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 3 \end{pmatrix}}_{U}$$

Вычисление LUP-разложения

В общем случае при решении системы линейных уравнений Ax = b может оказаться, что необходимо выбирать ведущие элементы среди недиагональных элементов матрицы A для того, чтобы избежать деления на 0. Причем нежелательно не только деление на 0, но и просто на малое число, даже если матрица A невырождена, поскольку в результате можно получить численную неустойчивость. В связи с этим в качестве ведущего элемента следует выбирать наибольший возможный элемент.

Математические основы LUP-разложения аналогичны LU-разложению. Вспомним, что имеется невырожденная матрица A размером n x n и требуется найти матрицу перестановки P, единичную нижнетреугольную матрицу L и верхнетреугольную матрицу U, такие, что PA = LU. Перед разделением матрицы A на части, как при вычислении LU-разложения, мы перемещаем ненулевой элемент, скажем, a_{k1}, откуда-то из первого столбца матрицы в позицию (1, 1) (если первый столбец содержит только нулевые значения, то матрица вырождена, поскольку ее определитель равен 0. Для сохранения множества исходных линейных уравнений мы меняем местами строки 1 и k, что эквивалентно умножению матрицы A на матрицу перестановки Q слева.

Вычисление LUP-разложения

Тогда мы можем записать QA как

$$QA = \begin{pmatrix} a_{k1} & w^T \\ v & A' \end{pmatrix}$$

где v = $(a_{21}, a_{31}, \ldots, a_{n1})$, за исключением того, что a_{11} заменяет a_{k1}; w= $(a_{k2}, a_{k3}, \ldots, a_{kn})^T$; а A является матрицей размером (n - 1) × (n - 1). Поскольку $a_{k1} \neq 0$, можно выполнить почти те же преобразования, что и при LU-разложении, но теперь с гарантией, что деления на нуль не будет:

$$QA = \begin{pmatrix} a_{k1} & w^T \\ v & A' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ v/a_{k1} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & A' - vw^T/a_{k1} \end{pmatrix}$$

Как мы видели в LU-разложении, если матрица A невырождена, то невырождено и дополнение Шура $A' - vw^T/a_{k1}$. Таким образом, мы можем индуктивно найти его LUP-разложение, дающее единичную нижнетреугольную матрицу L', верхнетреугольную матрицу U' и матрицу перестановки P', такие, что

$$P'(A' - vw^T/a_{k1}) = L'U'$$

Вычисление LUP-разложения

Определим матрицу которая является матрицей перестановки в силу того, что она представляет собой произведение двух матриц перестановки. Мы имеем

$$PA = \begin{pmatrix} 1 & 0 \\ 0 & P' \end{pmatrix} QA = \begin{pmatrix} 1 & 0 \\ 0 & P' \end{pmatrix} \begin{pmatrix} 1 & 0 \\ v/a_{k1} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & A' - vw^T/a_{k1} \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 \\ P'v/a_{k1} & P' \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & A' - vw^T/a_{k1} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ P'v/a_{k1} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & P'(A' - vw^T/a_{k1}) \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 \\ P'v/a_{k1} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & L'U' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ P'v/a_{k1} & L' \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & U' \end{pmatrix} = LU,$$

что и дает искомое LUP-разложение. Поскольку L' - единичная нижнетреугольная матрица, такой же будет и L, и поскольку U' - верхнетреугольная матрица, такой же будет и U. Заметим, что в отличие от LU-разложения, на матрицу перестановки P' должны умножаться и вектор-столбец v/ak1, и дополнение Шура A' - vw /ak1. Вот как выглядит псевдокод LUP-разложения.

Вычисление LUP-разложения

```
LUP-Decomposition(A)
n = A.rows
π[1…n] - вновь созданный массив
for i = 1 to n
    π[i] = i
for k = 1 to n
    p = 0
    for i = k to n
        if |a_ik| > p
            p = |atk|
            k’ = i
    if p == 0
        error "вырожденная матрица"
    Обменять π[k] с π[k’]
    for i = 1 to n
        Обменять a_ki с a_k’i
    for i = k + 1 to n
        a_ik = a_ik/a_kk
        for j = k + 1 to n
            a_ij = a_ij - a_ik a_kj
```

Вычисление LUP-разложения

Как и в процедуре LUP-Decomposition, в псевдокоде LUP-разложения LUP-Decomposition рекурсия заменяется итерацией. В качестве усовершенствования непосредственной реализации рассмотренной рекурсии мы динамически поддерживаем матрицу перестановки P в виде массива π, где π[i] = j означает, что i-я строка массива P содержит 1 в столбце j. Мы также реализуем код, который вычисляет L и U "на месте", в матрице A, т.е. по окончании работы процедуры

$$a_{ij} = \begin{cases} l_{ij}, & \text{если } i > j \\ u_{ij}, & \text{если } i \leq j \end{cases}$$

В силу тройной вложенности циклов время работы процедуры LUP-Decomposition равно $\Theta(n^3)$, т.е. оно точно такое же, как и в случае процедуры LU- Decomposition. Следовательно, выбор ведущего элемента приводит увеличению времени работы не более чем на постоянный множитель.

Вычисление LUP-разложения

1	2	0	2	0.6
2	3	3	4	-2
3	**5**	5	4	2
4	-1	-2	3.4	-1

3	**5**	5	4	2
2	3	3	4	-2
1	2	0	2	0.6
4	-1	-2	3.4	-1

3	**5**	5	4	2
2	0.6	0	1.6	-3.2
1	0.4	-2	1.4	-2
4	-0.2	-1	4.2	-0.6

3	5	5	4	2
2	0.6	0	1.6	-3.2
1	0.4	**-2**	1.4	-2
4	-0.2	-1	4.2	-0.6

3	5	5	4	2
1	0.4	**-2**	1.4	-2
2	0.6	0	1.6	-3.2
4	-0.2	-1	4.2	-0.6

3	5	5	4	2
1	0.4	**-2**	1.4	-2
2	0.6	0	1.6	-3.2
4	-0.2	0.5	4	-0.5

3	5	5	4	2
1	0.4	-2	1.4	-2
2	0.6	0	1.6	-3.2
4	-0.2	0.5	**4**	-0.5

3	5	5	4	2
1	0.4	-2	1.4	-2
4	-0.2	0.5	**4**	-0.5
2	0.6	0	1.6	-3.2

3	5	5	4	2
1	0.4	-2	1.4	-2
4	-0.2	0.5	**4**	-0.5
2	0.6	0	0.4	-3

$$\underset{P}{\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}} \underset{A}{\begin{pmatrix} 2 & 0 & 2 & 0.6 \\ 3 & 3 & 4 & -2 \\ 5 & 5 & 4 & 2 \\ -1 & -2 & 3.4 & -1 \end{pmatrix}} = \underset{L}{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.4 & 1 & 0 & 0 \\ -0.2 & 0.5 & 1 & 0 \\ 0.6 & 0 & 0.4 & 1 \end{pmatrix}} \underset{U}{\begin{pmatrix} 5 & 5 & 4 & 2 \\ 0 & -2 & 1.4 & -2 \\ 0 & 0 & 4 & -0.5 \\ 0 & 0 & 0 & -3 \end{pmatrix}}$$

Обращение матриц

Хотя на практике для решения систем линейных уравнений обращение матриц обычно не используется, а вместо этого применяются другие, численно более устойчивые, методы, например LUP-разложение, иногда все же требуется вычислить обратную матрицу. В этом разделе мы покажем, что для обращения матриц можно использовать уже рассмотренный нами метод LUP-разложения. Мы также докажем, что умножение матриц и обращение матрицы - задачи одинаковой сложности, так что для решения одной задачи мы можем использовать алгоритм для решения другой, получив при этом одинаковое асимптотическое время работы. В частности, для обращения матриц мы можем применить алгоритм Штрассена

Вычисление обратной матрицы из LUP-разложения

Предположим, что имеется LUP-разложение матрицы A на три матрицы, L, U и P, такие, что PA = LU. Используя процедуру LUP-Solve, мы можем решить уравнение вида Ax = b за время $\Theta(n^2)$. Поскольку LUP-разложение зависит только от A, но не от b, мы можем использовать ту же процедуру LUP-SOLVE для решения другой системы линейных уравнений вида Ax = b' за дополнительное время $\Theta(n^2)$. Обобщая, имея LUP-разложение матрицы A, мы можем решить k систем линейных уравнений Ax = b, отличающихся только свободными членами b, за время $\Theta(kn^2)$. Уравнение

$$AX = I_n$$

определяющее матрицу X, обратную A, можно рассматривать как множество из n различных систем линейных уравнений вида Ax = b. Эти уравнения позволяют найти матрицу X, обратную матрице A. Более строго, обозначим i-й столбец Характеристики через X_i и вспомним, что i-м столбцом матрицы I_n является единичный вектор e;. Мы можем найти X в уравнении, использовав LUP-разложение для решения набора уравнений

$$AX_i = e_i$$

для каждого X_i в отдельности. При наличии LUP-разложения поиск каждого столбца X_i требует времени $\Theta(n^2)$, так что полное время вычисления обратной матрицы X на основе LUP-разложения исходной матрицы A требует времени $\Theta(n^3)$.

Умножение матриц и обращение матрицы

Теперь покажем, каким образом можно использовать ускоренное умножение матриц для ускорения обращения матрицы (это ускорение имеет скорее теоретический интерес, чем практическое применение). В действительности мы докажем более строгое утверждение - умножение матриц эквивалентно обращению матрицы в следующем смысле. Если обозначить через M(n) время, необходимое для умножения двух матриц размером n x n, то можно обратить невырожденную матрицу размером n × n за время O(M(n)). Кроме того, если обозначить через I(n) время, необходимое для обращения матрицы размером n × n, то можно перемножить две матрицы размером n п за время O(I(n)).

Умножение матриц и обращение матрицы

Теорема (Умножение не сложнее обращения)
Если можно обратить матрицу размером n × n за время I(n), где $I(n) = \Omega(n^2)$ и удовлетворяет условию регулярности I(3n) = O(I(n)), то две матрицы размером n × n можно перемножить за время O(I(n)).

Доказательство. Пусть A и B представляют собой матрицы размером n × n, произведение которых C необходимо вычислить. Определим матрицу D размером 3n × 3n как

$$D = \begin{pmatrix} I_n & A & 0 \\ 0 & I_n & B \\ 0 & 0 & I_n \end{pmatrix}.$$

Обратной к матрице D является

$$D^{-1} = \begin{pmatrix} I_n & -A & AB \\ 0 & I_n & -B \\ 0 & 0 & I_n \end{pmatrix},$$

так что мы можем вычислить произведение AB, просто взяв верхнюю правую подматрицу размером n × n из матрицы D^{-1}. Матрицу D можно построить за время $\Theta(n^2)$, которое представляет собой O(I(n)), поскольку мы считаем, что $I(n) = \Omega(n^2)$, и обратить ее за время O(I(3n)) = O(I(n)) согласно условию регулярности, накладываемому на I(n). Таким образом, M(n) = O(I(n)). Заметим, что $I(n) = \Theta(n^c \lg^d n)$ удовлетворяет условию регулярности при любых константах c > 0 и d > 0.

Умножение матриц и обращение матрицы

Теорема (Обращение не сложнее умножения)

Предположим, что мы можем умножить две действительные матрицы размером n n за время M(n), где M(n) = Ω(n^2) и, кроме того, удовлетворяет условиям регулярности M(n + k) = O(M(n)) для произвольного 0 ≤ k ≤ n и M(n/2) ≤ cM(n) для некоторой константы c < 1/2. В таком случае мы можем обратить любую действительную невырожденную матрицу размером n × n за время O(M(n)).

Доказательство. Здесь данная теорема доказывается для действительных чисел.

Можно считать, что n является точной степенью 2, поскольку мы имеем

$$\begin{pmatrix} A & 0 \\ 0 & I_k \end{pmatrix}^{-1} = \begin{pmatrix} A^{-1} & 0 \\ 0 & I_k \end{pmatrix}$$

для любого k > 0. Таким образом, выбрав k так, чтобы величина n + k была степенью 2, мы увеличиваем исходную матрицу до размера, представляющего собой степень 2, а искомую обратную матрицу A^{-1} получаем как часть обращенной матрицы большего размера. Первое условие регулярности M(n) гарантирует, что такое увеличение не вызовет увеличения времени работы более чем на постоянный множитель.

Предположим теперь, что матрица A имеет размер n x n и является симметричной и положительно определенной.

Умножение матриц и обращение матрицы

Разобьем матрицу A и обратную к ней A^{-1} на четыре подматрицы размером n/2 × n/2:

$$A = \begin{pmatrix} B & C^T \\ C & D \end{pmatrix} \text{ и } A^{-1} = \begin{pmatrix} R & T \\ U & V \end{pmatrix}$$

Обозначив через

$$S = D - CB^{-1}C^T,$$

дополнение Шура матрицы A по отношению к подматрице B, имеем

$$A^{-1} = \begin{pmatrix} R & T \\ U & V \end{pmatrix} = \begin{pmatrix} B^{-1} + B^{-1}C^TS^{-1}CB^{-1} & -B^{-1}C^TS^{-1} \\ -S^{-1}CB^{-1} & S^{-1} \end{pmatrix},$$

поскольку $AA^{-1} = I_n$, как вы можете убедиться, перемножив матрицы. Матрица A симметрична и положительно определена, что и B, и S симметричны и положительно определены. Таким образом, существуют обратные матрицы B^{-1} и S^{-1}, а B^{-1} и S^{-1} симметричны, так что $(B^{-1})^T = B^{-1}$ и $(S^{-1})^T = S^{-1}$. Итак, мы можем вычислить подматрицы R, T, U и V матрицы A^{-1} описанным далее способом.

1. Образуем подматрицы B, C, C^T и D матрицы A.

2. Рекурсивно вычислим обратную матрице B матрицу B^{-1}.

3. Вычислим произведение матриц $W = CB^{-1}$, а затем транспонируем его W^T, получив матрицу, эквивалентную $B^{-1}C^T$ ($(B^{-1})^T = B^{-1}$).

4. Вычислим произведение матриц $X = WC^T$, эквивалентное $CB^{-1}C^T$, затем - матрицу $S = D - X = D - CB^{-1}C^T$.

5. Рекурсивно вычислим обратную матрицу S-1 и присвоим ее матрице V.

6. Вычислим произведение матриц $Y = S^{-1}W$, равное $S^{-1}CB^{-1}$, а затем транспонируем его Y^T, что равно $B^{-1}C^TS^{-1}$ ($(B^{-1})^T= B^{-1}$ и $(S^{-1})^T = S^{-1}$). Присвоим T значение $-Y^T$, а U - значение -Y.

7. Вычислим произведение матриц $Z = W^TY$, равное $B^{-1}C^TS^{-1}CB^{-1}$, и присвоим R значение $B^{-1} + Z$.

Умножение матриц и обращение матрицы

Таким образом, можно инвертировать симметричную положительно определенную матрицу размером n × n путем обращения двух матриц размером n/2 × n/2, последующего выполнения четырех перемножений матриц размером n/2 × n/2, а также выполнения дополнительных действий по извлечению подматриц из A, вставке подматриц в A^{-1} ценой $O(n^2)$, и константного количества сложений, вычитаний и транспонирований матриц размером п/2×n/2. В результате мы получаем следующее рекуррентное соотношение:

$$I(n) \leq 2I\left(\frac{n}{2}\right) + 4M\left(\frac{n}{2}\right) + O(n^2) = 2I(n/2) + \Theta(M(n)) = O(M(n)).$$

Вторая строка выполняется, поскольку из второго условия регулярности в формулировке теоремы вытекает, что $4M(n/2) < 2M(n)$, и поскольку мы предполагаем, что $M(n) = \Omega(n^2)$.

Остается доказать, что асимптотическое время умножения матриц может быть получено для обращения невырожденной матрицы A, которая не является симметричной и положительно определенной. Основная идея заключается в том, что для любой невырожденной матрицы A матрица A^TA и положительно определенная. Все, что остается, — это привести задачу обращения матрицы A к задаче обращения матрицы A^TA.

Такое приведение основано на наблюдении, что если A является невырожденной матрицей размером n × n, то $A^{-1} = (A^TA)^{-1}A^T$, поскольку $((A^TA)^{-1}A^T)A = (A^TA)^{-1}(A^TA) = I_n$, а обратная матрица единственна. Следовательно, можно вычислить A^{-1}, сначала умножив A^T на A для получения симметричной положительно определенной матрицы A^TA, а затем обратив эту матрицу с помощью описанного выше рекурсивного алгоритма и умножив полученный результат на A^T. Каждый из перечисленных шагов требует времени O(M(n)), так что обращение любой невырожденной матрицы с действительными элементами может быть выполнено за время O(M(n)).

Симметричные положительно определенные матрицы и метод наименьших квадратов

Лемма Любая симметричная положительно определенная матрица является невырожденной.

Доказательство. Предположим, что матрица A вырождена. Тогда имеется такой ненулевой вектор x, что Ax = 0. Следовательно, $x^T Ax = 0$, и A не может быть положительно определенной.

Доказательство того факта, что LU-разложение симметричных положительно определенных матриц можно выполнить, не опасаясь столкнуться с делением на 0, более сложное. Начнем с доказательства свойств некоторых определенных подматриц A. Определим k-ю ***главную подматрицу*** (leading submatrix) A как матрицу A_k, состоящую из пересечения к первых строк и k первых столбцов A.

Симметричные положительно определенные матрицы и метод наименьших квадратов

Лемма Если A представляет собой симметричную положительно определенную матрицу, то все ее главные подматрицы симметричные и положительно определенные.

Доказательство. То, что каждая главная подматрица A_k является симметричной, очевидно. Для доказательства того, что она положительно определенная, воспользуемся методом от противного. Если не является положительно определенной, то существует вектор $x_k \neq 0$ размером k, такой, что $x_k^T A_k x_k$. Пусть матрица A имеет размер n × n и

$$A = \begin{pmatrix} A_k & B^T \\ B & C \end{pmatrix}$$

с подматрицами B (размером (n - k) × k) и C (размером (n - k) × (n - k)). Определим вектор x = (x_k^T 0)^T размером n, в котором после x следуют n – k нулей. Тогда

$$x_k^T A x = \begin{pmatrix} x_k^T & 0 \end{pmatrix} \begin{pmatrix} A_k & B^T \\ B & C \end{pmatrix} \begin{pmatrix} x_k \\ 0 \end{pmatrix} = \begin{pmatrix} x_k^T & 0 \end{pmatrix} \begin{pmatrix} A_k x_k \\ B x_k \end{pmatrix} = x_k^T A_k x_k \leq 0$$

что противоречит условию, что матрица A положительно определенная.

Симметричные положительно определенные матрицы и метод наименьших квадратов

Лемма (Лемма о дополнении Шура) Если A представляет собой симметричную положительно определенную матрицу, а A_k - главная подматрица A размером k, то дополнение Шура S матрицы A относительно подматрицы Ак является симметричным положительно определенным.

Доказательство. Поскольку матрица A симметрична, симметрична также подматрица C. Дополнение Шура S также является симметричным.

Остается показать, что дополнение Шура S положительно определенное. Для любого ненулевого вектора x в соответствии с предположением о том, что A является положительно определенной матрицей, выполняется соотношение $x^T Ax > 0$. Разобьем x на два подвектора, y и z, совместимые с Ак и C соответственно. В силу существования A имеем

$$x^T Ax = \begin{pmatrix} y^T & z^T \end{pmatrix} \begin{pmatrix} A_k & B^T \\ B & C \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} y^T & z^T \end{pmatrix} \begin{pmatrix} A_k y + B^T z \\ By + Cz \end{pmatrix}$$

$$= y^T A_k y + y^T B^T z + z^T By + z^T Cz$$

$$= \left(y + A_k^{-1} B^T z\right)^T A_k \left(y + A_k^{-1} B^T z\right) + z^T \left(C - BA_k^{-1} B^T\right) z.$$

Поскольку неравенство $x^T Ax > 0$; $z^T\left(C - BA_k^{-1}B^T\right)z = z^T Sz = x^T Ax > 0$.

Метод наименьших квадратов

Одним из важных приложений симметричных положительно определенных матриц является подбор кривой для заданного множества экспериментальных точек. Предположим, что дано множество из m точек

(x_1,y_1), (x_2, y_2),… ,(x_m,y_m),

где значения y, содержат ошибки измерений. Нужно найти функцию F(x), такую, что ошибки аппроксимации

$$\eta_i = F(x_i) - y_i$$

малы при i = 1,2,…,m. Вид функции F зависит от рассматриваемой задачи, и здесь мы будем считать, что она имеет вид линейной взвешенной суммы

$$F(x) = \sum_{j=1}^{n} c_j f_j(x),$$

где количество слагаемых n и набор базисных функций (basis functions) f выбираются на основе знаний о рассматриваемой задаче. Зачастую в качестве базисных функций выбираются $f_j(x) = x^{j-1}$, т.е. функция F представляет собой полином степени n - 1 от x:

$$F(x) = c_1 + c_2 x + c_3 x^2 + \cdots + c_n x^{n-1}.$$

Таким образом, для заданных т экспериментальных точек (x_1,y_1), (x_2, y_2),… ,(x_m,y_m) необходимо вычислить п коэффициентов c_1, c_2,…, c_n, минимизирующих ошибки приближения $\eta_1, \eta_2, \dots, \eta_m$.

Выбрав n = m, можно точно вычислить все y_i.

Метод наименьших квадратов

Выбор такой функции F с высокой степенью не слишком удачен, так как, помимо данных, он учитывает и весь "шум", что приводит к плохим результатам при использовании F для предсказания значения y для некоторого x, измерения для которого еще не выполнялись. В любом случае, когда выбрано n, меньшее, чем m, мы получаем переопределенную систему линейных уравнений, приближенное решение которой хотим найти. Пусть

$$A = \begin{pmatrix} f_1(x_1) & f_2(x_1) & \cdots & f_n(x_1) \\ f_1(x_2) & f_2(x_2) & \cdots & f_n(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(x_m) & f_2(x_m) & \cdots & f_n(x_m) \end{pmatrix}$$

обозначает матрицу значений базисных функций в заданных точках, т.е. $a_{ij} = f_i(x_j)$, и пусть c = (c_k) обозначает искомый вектор коэффициентов размером n.

Тогда

$$Ac = \begin{pmatrix} f_1(x_1) & f_2(x_1) & \cdots & f_n(x_1) \\ f_1(x_2) & f_2(x_2) & \cdots & f_n(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(x_m) & f_2(x_m) & \cdots & f_n(x_m) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} F(x_1) \\ F(x_2) \\ \vdots \\ F(x_n) \end{pmatrix}$$

представляет собой вектор размером m "предсказанных значений" y, а вектор

$$\eta = Ac - y$$

является вектором невязок (ошибок приближения - approximation error) размером m.

Метод наименьших квадратов

Для минимизации невязок будем минимизировать норму вектора ошибок η, что отражено в названии "решение методом наименьших квадратов" (least-squa resolution), так как

$$\left|\left|\eta\right|\right| = \left(\sum_{i=1}^{m} \eta_i^2\right)^{1/2}$$

Поскольку

$$\left|\left|\eta\right|\right|^2 = \left|\left|Ac - y\right|\right|^2 = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} c_j - y_i\right)^2 ,$$

можно минимизировать $\left|\left|\eta\right|\right|^2$, дифференцировав $\left|\left|\eta\right|\right|^2$ по всем c_k и приравняв полученные производные к 0:

$$\frac{d\left|\left|\eta\right|\right|^2}{dc_k} = \sum_{i=1}^{m} 2\left(\sum_{j=1}^{n} a_{ij} c_j - y_i\right) a_{ik} = 0.$$

n уравнений (2.18) с k = 1, 2, …, n эквивалентны одному матричному уравнению

$(Ac - y)^T A = 0$,

или, что то же самое,

$A^T(Ac - y) = 0$,

откуда вытекает

$$A^T Ac = A^T y.$$

Метод наименьших квадратов

В качестве примера рассмотрим пять экспериментальных точек,
$(x_1,y_1) = (-1,2)$, $(x_2,y_2) = (1,1)$, $(x_3, y_3) = (2,1)$, $(x_4,y_4) = (3,0)$, $(x_5, y_5) = (5,3)$,
Необходимо найти приближение экспериментальных данных квадратичным полиномом
$F(x) = c_1 + c_2x + c_3x^2$.
Начнем с матрицы значений базисных функций:

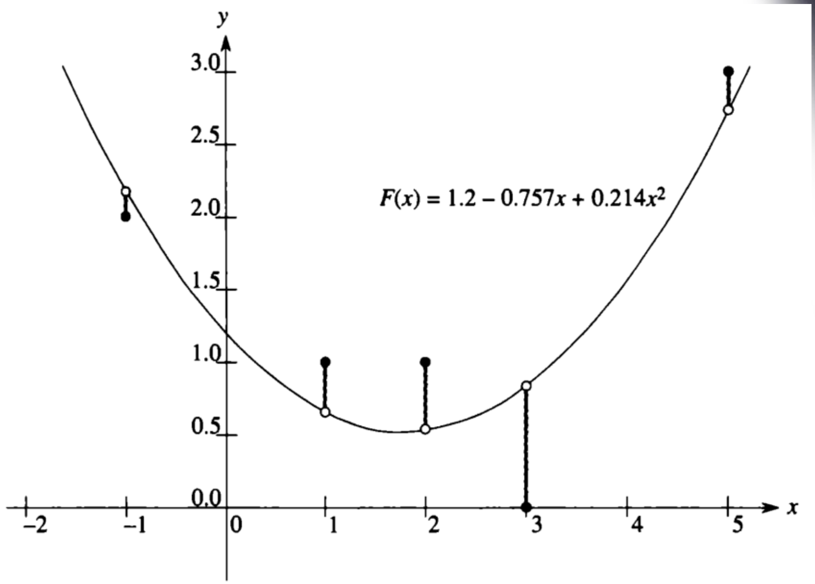
$$A = \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \\ 1 & x_4 & x_4^2 \\ 1 & x_5 & x_1^2 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 1 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 5 & 25 \end{pmatrix}.$$

Ее псевдообратная матрица имеет вид

$$A^+ = \begin{pmatrix} 0.500 & 0.300 & 0.200 & 0.100 & -0.100 \\ -0.388 & 0.093 & 0.190 & 0.193 & -0.088 \\ 0.060 & -0.036 & -0.048 & -0.036 & 0.060 \end{pmatrix}$$

Умножив y на A^+, получаем вектор коэффициентов

$$c = \begin{pmatrix} 1.200 \\ -0.757 \\ 0.214 \end{pmatrix}$$



$F(x) = 1.200 - 0.757x + 0.214x^2$, который представляет собой наилучшее квадратичное приближение экспериментальных данных.

Итерационные методы

Сложность $\mathcal{O}(n^2 t)$, t – число шагов.
Пусть дана система линейных алгебраических уравнений:
Ax = *F*
Выбирается начальное приближение $x^{(0)} = \left(x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)}\right)$ (при отсутствии априорных данных для выбора приближения в качестве начального приближения можно выбрать нулевой вектор).
Метод Якоби. Каждое следующее приближение в методе Якоби рассчитывается по формуле:

$$x_i^{(k+1)} = x_i^{(k)} + \frac{1}{a_{ii}}\left[f_i - \sum_{j=1}^{n} a_{ij} x_j^{(k)}\right], i = 1, \dots, n$$

где *k* – номер текущей итерации.
Метод Гаусса-Зейделя. Каждое последующее приближение рассчитывается по формуле:

$$x_i^{(k+1)} = x_i^{(k)} + \frac{1}{a_{ii}}\left[f_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij} x_j^{(k)}\right], i = 1, \dots, n$$

Для ускорения сходимости итерационного процесса можно использовать *параметр релаксации*.

Итерационные методы

Итерационный процесс в ***методе Якоби с параметром релаксации*** выглядит следующим образом:

$$\hat{x}_i^{(k+1)} = x_i^{(k)} + \frac{1}{a_{ii}}\left[f_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij}x_j^{(k)}\right],$$

$$x_i^{(k+1)} = w\hat{x}_i^{(k+1)} + (1-w)x_i^{(k)}$$

Подставляя в приближение в методе Якоби, получаем:

$$x_i^{(k+1)} = x_i^{(k)} + \frac{w}{a_{ii}}\left[f_i - \sum_{j=1}^{n} a_{ij}x_j^{(k)}\right], 0 < w \le 1$$

В методе ***Гаусса-Зейделя с параметром релаксации*** (другое название - ***метод релаксации***) итерационный процесс описывается следующим образом:

$$x_i^{(k+1)} = x_i^{(k)} + \frac{w}{a_{ii}}\left[f_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij}x_j^{(k)}\right], 0 < w < 2$$

Условия выхода из итерационного процесса для рассмотренных методов:

1. Выход по относительной невязке: $\frac{\|F - Ax^{(k)}\|}{\|F\|} < \varepsilon$
2. Защита от зацикливания: *k* > *maxiter*, *maxiter* - максимальное количество итераций.

Решение систем нелинейных уравнений методом Ньютона

Пусть дана СНУ в виде:

$$\begin{gathered} F_1(x_1, x_2, \dots, x_n) = 0; \\ F_2(x_1, x_2, \dots, x_n) = 0; \\ \vdots \\ F_m(x_1, x_2, \dots, x_n) = 0; \end{gathered}$$

Обозначим через x^k решение, полученное на k-й итерации процесса Ньютона (для первой итерации x^0 – начальное приближение). Запишем исходную систему в виде $F_1\left(x^k + \Delta x^k\right) = 0\,, i\ = 1\ldots m,$ где $\Delta x^k = \bar{x}\ -\ x^k$, $\bar{x}$ – искомое решение.

Выполним линеаризацию i-го уравнения системы с использованием его разложения в ряд Тейлора в окрестности точки x^k :

$$F_i(x) \approx F_i\left(x^k\right) + \sum_{j=1}^{n} \left.\frac{\partial F_i(x)}{\partial x_j}\right|_{x=x^k} \Delta x_j^k, i = 1 \dots m$$

или, в матричном виде:

$$A^k\,\Delta x^k = \ -F^k,$$

где F^k – значение вектор-функции F при x = x^k; A^k – матрица Якоби ($A_{ij}^k = \left.\frac{\partial F_i(x)}{\partial x_j}\right|_{x=x^k}$).

Решение систем нелинейных уравнений методом Ньютона

Это система уравнений, линейных относительно приращений Δx_j^k. Решив эту систему, найдем направление Δx^k поиска решения.

Для поиска следующего приближения x^{k+1} вдоль направления Δx^k организуем итерационный процесс:

$$x_v^{k+1} = x^k + \beta_v^k \Delta x^k,$$

где β_v^k – параметр итерационного процесса поиска x^{k+1}, ($0 < \beta^k < 1$), v – номер итерации поиска оптимального значения β^k. Параметр β^k будем искать следующим образом: сначала (то есть после нахождения направления Δx^k) β^k принимается равным 1 и вычисляется значение $F_v^k = F(x^k + \beta_v^k \Delta x^k)\, n$; далее, пока норма F_v^k больше, чем норма F^{k-1}, β_v^k уменьшается вдвое.

Заметим, что в СЛАУ матрица A^k при несовпадении числа неизвестных и числа уравнений становится прямоугольной. В этом случае вместо СЛАУ решают другую (измененную) СЛАУ с квадратной матрицей, решение которой является решением.

План лекции

Командная строка	Решение СЛАУ	Вычеты, КТО	Группы, кольца, поля (начало)
10 минут	55 минут	20 минут	5 минут

Полная и приведенная системы вычетов

Определение. Любое число из класса эквивалентности $\equiv_m$ будем называть вычетом по модулю m . Совокупность вычетов, взятых по одному из каждого класса эквивалентности $\equiv_m$, называется полной системой вычетов по модулю m (в полной
системе вычетов, таким образом, всего m штук чисел). Непосредственно сами остатки при делении на m называются наименьшими неотрицательными вычетами и, конечно, образуют полную систему вычетов по модулю m. Вычет ρ называется абсолютно наименьшим, если $|\rho|$ наименьший среди модулей вычетов данного класса.

Пример: Пусть $m = 5$. Тогда:

0, 1, 2, 3, 4 − наименьшие неотрицательные вычеты;

−2, −1, 0, 1, 2 − абсолютно наименьшие вычеты.

Обе приведенные совокупности чисел образуют полные системы вычетов по модулю 5.

Полная и приведенная системы вычетов

Лемма 1. 1) Любые m штук попарно не сравнимых по модулю m чисел образуют полную систему вычетов по модулю m.

2) Если a и m взаимно просты, а x пробегает полную систему вычетов по модулю m, то значения линейной формы ax + b , где b − любое целое число, тоже пробегают полную систему вычетов по модулю m .

Доказательство. Утверждение 1) – очевидно. Докажем утверждение 2). Чисел ax + b ровно m штук. Покажем, что они между собой не сравнимы по модулю m. Ну пусть для некоторых различных x_1 и x_2 из полной системы вычетов оказалось, что $ax_1 + b \equiv ax_2 + b \pmod m$ 1 2 + +. Тогда, по свойствам сравнений, получаем:

$ax_1 \equiv ax_2 \pmod m$

$x_1 \equiv x_2 \pmod m$

– противоречие с тем, что x_1 и x_2 различны и взяты из полной системы вычетов.

Поскольку все числа из данного класса эквивалентности $\equiv_m$ получаются из одного числа данного класса прибавлением числа, кратного m, то все числа из данного класса имеют с модулем m один и тот же наибольший общий делитель. По некоторым соображениям, повышенный интерес представляют те вычеты, которые имеют с модулем m наибольший общий делитель, равный единице, т.е. вычеты, которые взаимно просты с модулем.

Определение. Приведенной системой вычетов по модулю m называется совокупность всех вычетов из полной системы, взаимно простых с модулем m .

Пример. Пусть m = 42. Тогда приведенная система вычетов суть:

1, 5, 11, 13, 17, 19, 23, 25, 29, 31, 37, 41.

Введение в СС

Далее будем рассматривать и учиться решать сравнения с одним неизвестным вида:
$f(x) \equiv 0 (\mathrm{mod}\ m)$,
где $f(x) = a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n$ – многочлен с целыми коэффициентами.
Если m не делит a_0, то говорят, что n – степень сравнения. Ясно, что если какое-нибудь число x подходит в сравнение, то в это же сравнение подойдет и любое другое число, сравнимое с x по mod m.
Решить сравнение – значит найти все те x, которые удовлетворяют данному сравнению, при этом весь класс чисел по mod m считается за одно решение.
Таким образом, число решений сравнения есть число вычетов из полной системы, которые этому сравнению удовлетворяют.
Пример. Дано сравнение: $x^5 + x + 1 \equiv 0 (\mathrm{mod}\ 7)$.
Из чисел: 0, 1, 2, 3, 4, 5, 6, этому сравнению удовлетворяют два: $x_1 = 2$; $x_2 = 4$.
Это означает, что у данного сравнения два решения:
$x \equiv 2 (\mathrm{mod}\ 7)$ и $x \equiv 4 (\mathrm{mod}\ 7)$.

Сравнения первой степени

В этом пункте детально рассмотрим только сравнения первой степени вида ax ≡ b(mod m), оставив более высокие степени на съедение следующим пунктам. Как решать такое сравнение? Рассмотрим два случая.

Случай 1. Пусть a и m взаимно просты. Тогда несократимая дробь $\frac{m}{a}$ сама просится разложиться в цепную дробь:

$$\frac{m}{a} = q_1 + \cfrac{1}{q_2 + \cfrac{1}{q_3 + \cfrac{\ddots}{\cfrac{1}{q_n}}}}$$

Эта цепная дробь, разумеется, конечна, так как $\frac{m}{a}$ − рациональное число. Рассмотрим две ее последние подходящие дроби:

$$\delta_{n-1} = \frac{P_{n-1}}{Q_{n-1}};\ \delta_n = \frac{P_n}{Q_n} = \frac{m}{a}$$

Вспоминаем важное свойство числителей и знаменателей подходящих дробей: $mQ_{n-1} - aP_{n-1} = (-1)^n$. Далее (слагаемое mO_{n-1}, кратное m, можно выкинуть из левой части сравнения):

$$-aP_{n-1} \equiv (-1)^n \ (mod\ m)$$

$$aP_{n-1} \equiv (-1)^{n-1} \ (mod\ m)$$

$$-a\left[(-1)^{n-1} P_{n-1} b\right] \equiv b\ (mod\ m)$$

и единственное решение исходного сравнения есть:

$$x \equiv \left[(-1)^{n-1} P_{n-1} b\right] (mod\ m)$$

Сравнения первой степени

Пример. Решить сравнение 111x ≡ 75(mod 322) .

Решение. (111, 322) =1. Включаем алгоритм Евклида:

322=111·2+100

111=100·1+11

100=11·9+1

11=1·11

(В равенствах подчеркнуты неполные частные.) Значит, n = 4 , а соответствующая цепная дробь такова:

$$\frac{m}{a} = \frac{322}{111} = 2 + \cfrac{1}{1 + \cfrac{1}{9 + \cfrac{1}{11}}}$$

Посчитаем числители подходящих дробей, составив для этого стандартную таблицу:

q_n	0	2	1	9	11
P_n	1	2	3	29	322

Числитель предпоследней подходящей дроби равен 29, следовательно, готовая формула дает ответ: $x \equiv (-1)^3 \cdot 29 \cdot 75 \equiv -2175 \equiv 79 \pmod{322}$.

Сравнения первой степени

Случай 2. Пусть (a, m) = d. В этом случае, для разрешимости сравнения $ax \equiv b(\text{mod } m)$ необходимо, чтобы d делило b , иначе сравнение вообще выполняться не может. Действительно, $ax \equiv b(\text{mod } m)$ бывает тогда, и только тогда, когда ax – b делится на m нацело, т.е. $ax - b = t \cdot m$, $t \in Z$, откуда $b = ax - t \cdot m$, а правая часть последнего равенства кратна d.

Пусть $b = db_1$, $a = da_1$, $m = dm_1$. Тогда обе части сравнения $xa_1d \equiv b_1d(\text{mod } m_1d)$ и его модуль поделим на d:

$xa_1b \equiv (\text{mod } m_1)$, где уже a_1 и m_1 взаимно просты. Согласно случаю 1 этого пункта, такое сравнение имеет единственное решение x_0:

$x \equiv x_0 \ (\text{mod } m_1) \qquad\qquad (*)$

По исходному модулю m, числа $(*)$ образуют столько решений исходного сравнения, сколько чисел вида $(*)$ содержится в полной системе вычетов: $0, 1, 2, ..., m - 2, m - 1$.

Очевидно, что из чисел $x = x + t \cdot m$ в полную систему наименьших неотрицательных вычетов попадают только x_0, $x_0 + m_1$, $x_0 + 2m_1, \ldots, x_0 + (d - 1)\, m_1$, т.е. всего d чисел. Значит у исходного сравнения имеется d решений.

Подведем итог рассмотренных случаев в виде следующей теоремы

Сравнения первой степени

Случай 2. Пусть $(a, m) = d$. В этом случае, для разрешимости сравнения $ax \equiv b(\mathrm{mod}\ m)$ необходимо, чтобы d делило b , иначе сравнение вообще выполняться не может. Действительно, $ax \equiv b(\mathrm{mod}\ m)$ бывает тогда, и только тогда, когда $ax – b$ делится на m нацело, т.е. $ax - b = t \cdot m,\ t \in Z$, откуда $b = ax - t \cdot m$, а правая часть последнего равенства кратна d.

Пусть $b = db_1$, $a = da_1$, $m = dm_1$. Тогда обе части сравнения $xa_1d \equiv b_1d(\mathrm{mod}\ m_1d)$ и его модуль поделим на d:

$xa_1b \equiv (\mathrm{mod}\ m_1)$, где уже a_1 и m_1 взаимно просты. Согласно случаю 1 этого пункта, такое сравнение имеет единственное решение x_0:

$x \equiv x_0\ (\mathrm{mod}\ m_1) \qquad\qquad (*)$

По исходному модулю m, числа $(*)$ образуют столько решений исходного сравнения, сколько чисел вида $(*)$ содержится в полной системе вычетов: $0, 1, 2, ..., m - 2, m - 1$.

Очевидно, что из чисел $x = x + t \cdot m$ в полную систему наименьших неотрицательных вычетов попадают только x_0, $x_0 + m_1$, $x_0 + 2m_1, \ldots, x_0 + (d - 1)\, m_1$, т.е. всего d чисел. Значит у исходного сравнения имеется d решений.

Теорема. Пусть $(a, m) = d$. Если b не делится на d , сравнение $ax \equiv b(\mathrm{mod}\ m)$ не имеет решений. Если b кратно d, сравнение $ax \equiv b\ (\mathrm{mod}\ m)$ имеет d штук решений.

Сравнения первой степени

Пример. Решить сравнение $111x \equiv 75 \pmod{321}$.

Решение. $(111, 321) = 3$, поэтому поделим сравнение и его модуль на 3:

$37x \equiv 25 \pmod{107}$, и уже $(37,107) = 1$.

Включаем алгоритм Евклида (как обычно, подчеркнуты неполные частные):

$107 = 37 \cdot 2 + 33$

$37 = 33 \cdot 1 + 4$

$33 = 4 \cdot 8 + 1$

$4 = 1 \cdot 4$

Имеем $n = 4$ и цепная дробь такова:

$$\frac{m}{a} = \frac{107}{37} = 2 + \cfrac{1}{1 + \cfrac{1}{8 + \cfrac{1}{4}}}$$

Таблица для нахождения числителей подходящих дробей:

q_n	0	2	1	8	4
P_n	1	2	3	26	107

Значит, $x \equiv (-1)^3 \cdot 26 \cdot 25 \equiv -650 \pmod{107} \equiv -8 \pmod{107} \equiv 99 \pmod{107}$.

Три решения исходного сравнения:

$x \equiv 99 \pmod{321}$, $x \equiv 206 \pmod{321}$, $x \equiv 313 \pmod{321}$,

и других решений нет.

Сравнения первой степени

Теорема. Пусть $m > 1$, $(a, m) = 1$. Тогда сравнение $ax \equiv b(\mathrm{mod}\ m)$ имеет решение: $x \equiv ba^{\phi(m)-1}\ (\mathrm{mod}\ m)$.

Доказательство. По теореме Эйлера, имеем: $a^{\phi(m)} \equiv 1(\mathrm{mod}\ m)$, следовательно, $a \cdot ba^{\phi(m)-1} \equiv b(\mathrm{mod}\ m)$. ♦

Пример. Решить сравнение $7x \equiv 3(\mathrm{mod}\ 10)$. Вычисляем:

$\phi\ (10) = 4;\ x \equiv 3 \cdot 7^{4-1}\ (\mathrm{mod}\ 10) \equiv 1029(\mathrm{mod}\ 10) \equiv 9(\mathrm{mod}\ 10)$.

Видно, что этот способ решения сравнений хорош (в смысле минимума интеллектуальных затрат на его осуществление), но может потребовать возведения числа a в довольно большую степень, что довольно трудоемко. Для того, чтобы как следует это прочувствовать, возведите самостоятельно число 24789 в степень 46728.

Сравнения первой степени

Теорема. Пусть p – простое число, 0 < a < p . Тогда сравнение ax ≡ b (mod p) имеет решение:

$$x \equiv b(-1)^{a-1} \cdot \frac{(p-1)(p-2)\dots(p-a+1)}{1 \cdot 2 \cdot 3 \cdot \dots \cdot a} (\mathrm{mod}\ p) \equiv \mathrm{b}(-1)^{a-1} \cdot \frac{(p-1)!}{a!\,(p-a)!} (\mathrm{mod}\ p)$$

$$\equiv \mathrm{b}(-1)^{a-1} \cdot \frac{1}{p} \cdot C_p^a (\mathrm{mod}\ p)$$

где C_p^a – биномиальный коэффициент.

Доказательство непосредственно следует из очевидного сравнения

$$1 \cdot 2 \cdot 3 \cdot \dots \cdot a \cdot b(-1)^{a-1} \cdot \frac{(p-1)(p-2)\dots(p-a+1)}{1 \cdot 2 \cdot 3 \cdot \dots \cdot a} \equiv \mathrm{b} \cdot 1 \cdot 2 \cdot 3 \cdot \dots \cdot (a-1) (\mathrm{mod}\ p)$$

которое нужно почленно поделить на взаимно простое с модулем число 1· 2 · 3·...·(a − 1). ♦

Пример. Решить сравнение 7x ≡ 2(mod 11). Вычисляем:

$$C_{11}^7 = \frac{11!}{7!\,(11-7)!} = \frac{8 \cdot 9 \cdot 10 \cdot 11}{2 \cdot 3 \cdot 4} = 2 \cdot 3 \cdot 5 \cdot 11 = 330$$

Сравнения первой степени

Лемма 1 (Китайская теорема об остатках). Пусть дана простейшая система сравнений первой степени:

$$\begin{cases} x = b_1 (mod\ m_1) \\ x = b_2 (mod\ m_2) \\ \vdots \\ x = b_k (mod\ m_k) \end{cases}$$

где $m_1, m_2, \ldots, m_k$ попарно взаимно просты. Пусть, далее, $m_1 m_2 \ldots m_k = M_s m_s$; $M_s M_s^{\nabla} \equiv 1 (mod\ m_s)$ (Очевидно, что такое число M_s^{∇} всегда можно подобрать хотя бы с помощью алгоритма Евклида, т.к. $(m_s, M_s) = 1$); $x_0 = M_1 M_1^{\nabla} b_1 + M_2 M_2^{\nabla} b_2 + \cdots + M_k M_k^{\nabla} b_k$. Тогда система $(*)$ равносильна одному сравнению

$$x \equiv x_0 (mod\ m_1 m_2 \ldots m_k).$$

т.е. набор решений $(*)$ совпадает с набором решений сравнения

$$\begin{cases} x = x_0 (mod\ m_1) \\ x = x_0 (mod\ m_2) \\ \vdots \\ x = x_0 (mod\ m_k) \end{cases}$$

которая, очевидно, в свою очередь, равносильна одному сравнению $x \equiv x_0 (mod\ m_1 m_2 \ldots m_k)$.

Сравнения первой степени

Пример. Однажды средний товарищ подошел к умному товарищу и попросил его найти число, которое при делении на 4 дает в остатке 1, при делении на 5 дает в остатке 3, а при делении на 7 дает в остатке 2. Сам средний товарищ искал такое число уже две недели. Умный товарищ тут же составил систему:

$$\begin{cases} x \equiv 1(mod\ 4) \\ x \equiv 3(mod\ 5) \\ x \equiv 2(mod\ 7) \end{cases}$$

которую начал решать, пользуясь леммой 1. Вот его решение:
b_1 = 1, b_2 = 3, b_3 = 2 ; $m_1 m_2 m_3$ = 4 ·5· 7 = 4 · 35 = 5· 28 = 7 · 20 = 140 , т.е. M_1 = 35, M_2 = 28, M_3 = 20. Далее он нашел:

$$35 \cdot 3 \equiv 1(mod\ 4)$$
$$28 \cdot 2 \equiv 3(mod\ 5)$$
$$20 \cdot 6 \equiv 2(mod\ 7)$$

т.е. M_1^{∇} = 3, M_2^{∇} = 2, M_3^{∇} = 6 . Значит x_0 = 35· 3·1+ 28 · 2 · 3 + 20 · 6 · 2 = 513. После этого, по лемме 1, умный товарищ сразу получил ответ:
x ≡ 513(mod 140) ≡ 93(mod 140),
т.е. наименьшее положительное число, которое две недели искал средний товарищ, равно 93. Так умный товарищ в очередной раз помог среднему товарищу.
В следующей лемме, для краткости формулировки, сохранены обозначения леммы 1.

Сравнения первой степени

Лемма . Если $b_1 b_2 \dots b_k$ пробегают полные системы вычетов по модулям m_1, m_2,..., m_k соответственно, то x_0 пробегает полную систему вычетов по модулю $m_1 m_2 \dots m_k$.

Доказательство. Действительно, $x_0 = A_1 b_1 + A_2 b_2 + \cdots A_k b_k$ пробегает m_1, m_2,..., m_k различных значений. Покажем, что все они попарно не сравнимы по модулю $m_1 m_2 \dots m_k$.
Ну пусть оказалось, что

$$A_1 b_1 + A_2 b_2 + \cdots A_k b_k = A_1 b'_1 + A_2 b'_2 + \cdots A_k b'_k (mod\ m_1 m_2 \dots m_k)$$

Значит,

$$A_1 b_1 + A_2 b_2 + \cdots A_k b_k = A_1 b'_1 + A_2 b'_2 + \cdots A_k b'_k (mod\ m_s)$$

для каждого s, откуда

$$M_s M_s^{\nabla} b_s = M_s M_s^{\nabla} b'_s (mod\ m_s)$$

Вспомним теперь, что $M_s M_s^{\nabla} \equiv 1 (mod\ m_s)$, значит $M_s M_s^{\nabla} = 1 + m_s \cdot t$, откуда ($M_s M_s^{\nabla}$, m_s) = 1. Разделив теперь обе части сравнения

$$M_s M_s^{\nabla} b_s = M_s M_s^{\nabla} b'_s (mod\ m_s)$$

на число $M_s M_s^{\nabla}$, взаимно простое с модулем, получим, что $b_s \equiv b'_s (mod\ m_s)$, т.е. $b_s \equiv b'_s$ для каждого s.
Итак, x_0 пробегает $m_1 m_2 \dots m_k$ различных значений, попарно не сравнимых по модулю $m_1 m_2 \dots m_k$, т.е. полную систему вычетов.

Сравнения по простому модулю

рассмотрим сравнения вида $f(x) \equiv 0 (\mathrm{mod}\ p)$, где p −простое число,

$f(x)= ax^n + a_1x^{n-1} + \ldots + a_n$ − многочлен с целыми коэффициентами, и попытаемся научиться решать такие сравнения. Не отвлекаясь на посторонние природные явления, сразу приступим к работе.

Лемма 1. Произвольное сравнение $f(x) \equiv 0 (\mathrm{mod}\ p)$, где p − простое число, равносильно некоторому сравнению степени не выше $p - 1$.

Доказательство. Разделим $f(x)$ на многочлен $x^p - x$ (такой многочлен алгебраисты иногда называют “многочлен деления круга”) с остатком:

$f(x) = (x^p - x) \cdot Q(x) + R(x)$,

где, как известно, степень остатка $R(x)$ не превосходит $p - 1$. Но ведь, по теореме Ферма, $x\ p - x \equiv 0 (\mathrm{mod}\ p)$. Это означает, что $f(x) \equiv R(x) (\mathrm{mod}\ p)$, а исходное сравнение равносильно сравнению $R(x) \equiv 0 (\mathrm{mod}\ p)$. ♦

Сравнения по простому модулю

Лемма 2. Если сравнение $f(x)= ax^n + a_1x^{n-1} + \ldots + a_n \equiv 0 (\mathrm{mod}\ p)$ степени n по простому модулю p имеет более n различных решений, то все коэффициенты $a, a_1, \ldots, a_n$ кратны p .

Доказательство. Пусть сравнение $ax^n + a_1x^{n-1} + \ldots + a_n \equiv 0 (\mathrm{mod}\ p)$ имеет $n + 1$ решение и x_1, $x_2, \ldots, x_n$ – наименьшие неотрицательные вычеты этих решений. Тогда, очевидно, многочлен $f(x)$ представим в виде:

$$f(x) = a(x-x_1)(x-x_2)\ldots(x-x_{n-2})(x-x_{n-1})(x-x_n) + b(x-x_1)(x-x_2)\ldots(x-x_{n-2})(x-x_{n-1}) + c(x-x_1)(x-x_2)\ldots(x-x_{n-2}) + \cdots + k(x-x_1)(x-x_2) + l(x-x_1) + m$$

Действительно, коэффициент b нужно взять равным коэффициенту при x^{n-1} в разности $f(x) - a(x-x_1)(x-x_2)\ldots(x-x_{n-2})(x-x_{n-1})(x-x_n)$; коэффициент c – это коэффициент перед x^{n-2} в разности $f(x) - a(x-x_1)(x-x_2)\ldots(x-x_{n-2})(x-x_{n-1})(x-x_n) - b(x-x_1)(x-x_2)\ldots(x-x_{n-2})(x-x_{n-1})$ и т.д. Теперь положим последовательно $x =$ и $x_1, x_2, \ldots, x_n, x_{n+1}$. Имеем:

1) $f(x_1) = m \equiv 0 (\mathrm{mod}\ p)$, следовательно, p делит m .
2) $f(x_2) = m + l(x_2 - x_1) \equiv l(x_2 - x_1) \equiv 0 (\mathrm{mod}\ p)$, следовательно, p делит l, ибо p не может делить $x_2 - x_1$, так как $x_2 < p$, $x_1 < p$.
3) $f(x_3) \equiv l(x_3 - x_1)(x_3 - x_2) \equiv 0 (\mathrm{mod}\ p)$, следовательно, p делит k .

И т.д.

Подведем итог. ***Всякое нетривиальное сравнение по*** mod p ***равносильно сравнению степени не выше*** $p - 1$ ***и имеет не более*** $p - 1$ ***решений.***

Сравнения по простому модулю

Пример. $1 \cdot 2 \cdot 3 \cdot ... \cdot 10 + 1 = 3628800 + 1 = 3628801$ – делится на 11 (Вспомните признак делимости на 11 − если сумма цифр в десятичной записи числа на четных позициях совпадает с суммой цифр на нечетных позициях, то число кратно 11).

Пример-задача. Доказать, что если простое число p представимо в виде $4n+1$, то существует такое число x , что $x^2 + 1$ делится на p.

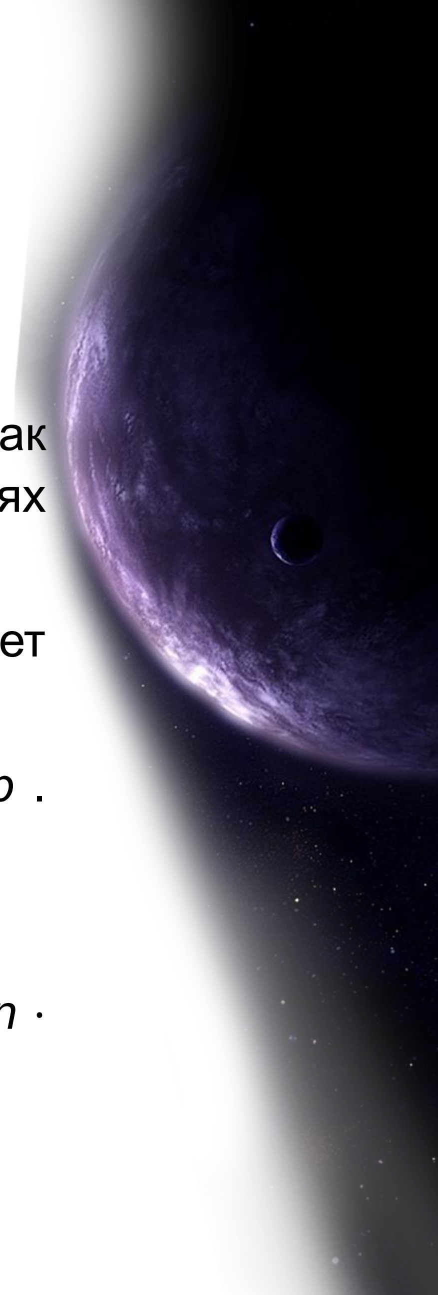
Решение. Пусть $p = 4n + 1$ – простое число. По теореме Вильсона, $(4n)! + 1$ делится на p . Заменим в выражении $1 \cdot 2 \cdot 3 \cdot ... \cdot (4n) + 1$ все множители большие $(p-1)/2 = 2n$

через разности числа p и чисел меньших $(p-1)/2 = 2n$. Получим:

$(p-1)!+1 = 1 \cdot 2 \cdot 3 \cdot ... \cdot 2n \cdot (p - 2n) \cdot (p - 2n - 1) \cdot \ldots \cdot (p - 1) = 1 \cdot 2 \cdot 3 \cdot ... \cdot 2n\, [A \cdot p + (-1)^{2n} \cdot 2n \cdot (2n-1) \cdot \ldots \cdot 2 \cdot 1] + 1 = A_1 \cdot p + (1 \cdot 2 \cdot 3 \cdot ... \cdot 2n)^2 + 1$

Так как это число делится на p , то и сумма $(1 \cdot 2 \cdot 3 \cdot ... \cdot 2n)^2+1$ делится на p , т.е.

$x = (2n)! = ((p-1)/2)!$



Сравнения по простому модулю

Теорема (Вильсон). Сравнение $(p-1)!+1\equiv 0 \pmod p$ выполняется тогда и только тогда, когда p − простое число.

Доказательство. Пусть p − простое число. Если $p = 2$, то, очевидно, $1!+1\equiv 0$

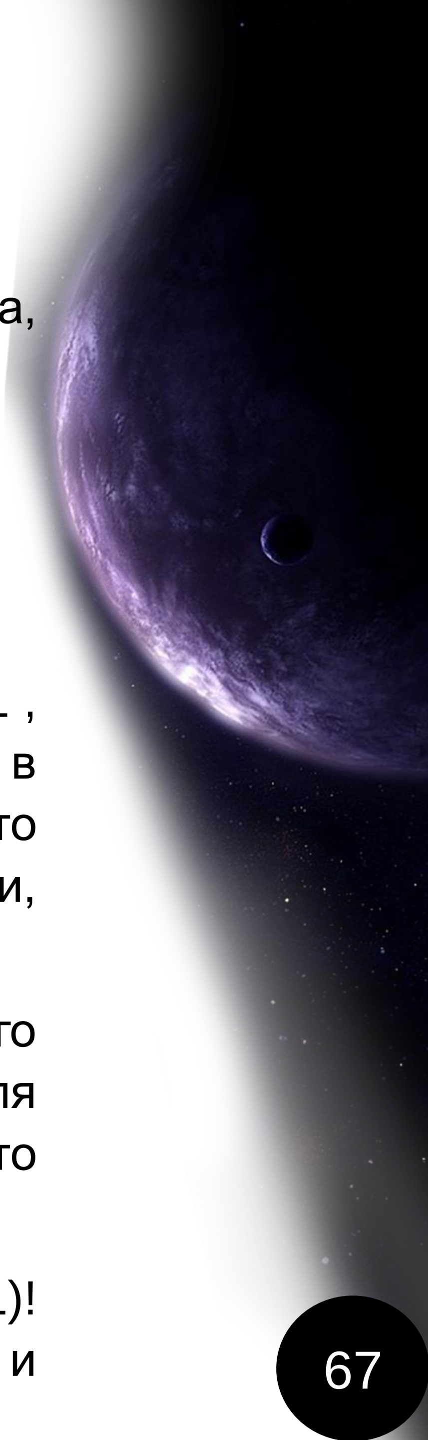
$\pmod 2$. Если $p > 2$, то рассмотрим сравнение:

$$[(x-1)(x-2)...(x-(p-1))]-(x^{p-1}-1)\equiv 0 \pmod p$$

Ясно, что это сравнение степени не выше $p-2$, но оно имеет $p-1$ решение: $1, 2, 3, \ldots, p-1$, т.к. при подстановке любого из этих чисел, слагаемое в квадратных скобках обращается в ноль, а $(x^{p-1}-1)$ сравнимо с нулем по теореме Ферма (x и p взаимно просты, т.к. $x < p$). Это означает, по лемме 2, что все коэффициенты выписанного сравнения кратны p, в частности, на p делится его свободный член, равный $1\cdot 2\cdot 3\cdot\ldots\cdot(p-1)+1$.

Так как коэффициенты многочлена являются значениями симметрических многочленов от его корней, то здесь наметился путь для доказательства огромного числа сравнений для симметрических многочленов. Однако, я по этому пути дальше не пойду, оставляя это прекрасное развлечение читателю, которому нечем коротать долгие зимние вечера.

Обратно. Если p – не простое, то найдется делитель d числа p, $1 < d < p$. Тогда $(p-1)!$ делится на d, поэтому $(p-1)!+1$ не может делится на d и, значит, не может делиться также и на p. Следовательно, сравнение $(p-1)!+1\equiv 0 \pmod p$ не выполняется. ♦



Сравнения по составному модулю

Теорема 1. Если числа m_1, m_2,..., m_kпопарно взаимно просты, то сравнение *f(x)* ≡ 0 (mod $m_1 m_2 \dots m_k$) равносильно системе сравнений:

$$\begin{cases} f(x) = 0(mod\ m_1) \\ f(x) = 0(mod\ m_2) \\ \vdots \\ f(x) = 0(mod\ m_k) \end{cases}$$

При этом, если T_1, T_2, ..., T_k – числа решений отдельных сравнений этой системы по соответствующим модулям, то число решений T исходного сравнения равно $T_1 T_2 \dots T_k$.

Доказательство. Первое утверждение теоремы (о равносильности системы и сравнения) очевидно, т.к. если $a \equiv b(\text{mod } m)$, то $a \equiv b(\text{mod } d)$, где d делит m . Если же $a \equiv b(\text{mod } m_1)$ и $a \equiv b(\text{mod } m_2)$, то $a \equiv b(\text{mod } НОК(m_1,m_2))$, где $НОК(m_1,m_2)$ – наименьшее общее кратное m_1 и m_2.

Обратимся ко второму утверждению теоремы (о числе решений сравнения).

Каждое сравнение $f(x) = 0(mod\ m_s)$ выполняется тогда и только тогда, когда выполняется одно из T_s штук сравнений вида $x = b_s(mod\ m_s)$, где b_s пробегает вычеты решений сравнения $f(x) = 0(mod\ m_s)$. Всего различных комбинаций таких простейших сравнений

$$\begin{cases} x = b_1(mod\ m_1) \\ x = b_2(mod\ m_2) \\ \vdots \\ x = b_k(mod\ m_k) \end{cases}$$

T_1T2 ...T_k штук. Все эти комбинации, приводят к различным классам вычетов по mod($m_1 m_2 \dots m_k$). ♦

Сравнения по составному модулю

Итак, решение сравнения $f(x) \equiv 0(\mathrm{mod}\ p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k})$ сводится к решениюсравнений вида $f(x) \equiv 0(\mathrm{mod}\ p^{\alpha})$. Оказывается, что решение этого последнего сравнения, в свою очередь, сводится к решению некоторого сравнения $g(x) \equiv 0(\mathrm{mod}\ p)$ с другим многочленом в левой части, но уже с простым модулем, а это, просто напросто, приводит нас в рамки предыдущего пункта. Сейчас я расскажу процесс сведения решения сравнения $f(x) \equiv 0(\mathrm{mod}\ p^{\alpha})$ к решению сравнения $g(x) \equiv 0(\mathrm{mod}\ p)$.

Процесс сведения.

Очевидно, выполнение сравнения $f(x) \equiv 0(\mathrm{mod}\ p^{\alpha})$ влечет, что x подходит в сравнение $f(x) \equiv 0(\mathrm{mod}\ p)$. Пусть $x \equiv x_1(\mathrm{mod}\ \mathrm{p})$ – какое-нибудь решение сравнения $f(x) \equiv 0(\mathrm{mod}\ p)$. Это означает, что $x = x_1 + p \cdot t_1$, где $t_1 \in \mathbf{Z}$.

Вставим это x в сравнение $f(x) \equiv 0(\mathrm{mod}\ p^2)$. Получим сравнение

$f(x_1 + p \cdot t_1)\ (\mathrm{mod}\ p^2)$,

которое тоже, очевидно, выполняется.

Разложим далее (не пугайтесь!) левую часть полученного сравнения по формуле Тейлора по степеням ($x - x^1$) :

$$\mathrm{f(x)} = \mathrm{f(x_1)} + \frac{\mathrm{f}'(\mathrm{x}_1)}{1!}(x - \mathrm{x}_1) + \frac{\mathrm{f}'(\mathrm{x}_1)^2}{2!}(x - \mathrm{x}_1)^2 + \cdots$$

Сравнения по составному модулю

Но, ведь, $x = x_1 + p \cdot t_1$, следовательно,

$$f(x_1 + p \cdot t_1) = f(x_1) + \frac{f'(x_1)}{1!}(p \cdot t_1) + \frac{f'(x_1)^2}{2!}(p \cdot t_1)^2 + \cdots$$

Заметим, что число

$\frac{f^{(k)}(x_1)}{k!}$ всегда целое, т.к. $f(x_1 + p \cdot t_1)$ – многочлен с целымикоэффициентами. Теперь в сравнении

$$f(x_1 + p \cdot t_1) \equiv 0 \ (mod\ p^2)$$

можно слева отбросить члены, кратные p^2:

$$f(x_1) + \frac{f'(x_1)}{1!}(pt_1) \equiv 0 \ (mod\ p^2)$$

Разделим последнее сравнение и его модуль на p :

$$\frac{f(x_1)}{p} + \frac{f'(x_1)}{1!}(t_1) \equiv 0 \ (mod\ p^2)$$

Заметим, опять, что

$\frac{f(x_1)}{p}$ – целое число, т.к. $f(x_1) \equiv 0 \ (mod\ p)$

Сравнения по составному модулю

Далее ограничимся случаем, когда значение производной $f'(x_1)$ не делится на p. В этом случае имеется всего одно решение сравнения первой степени $\frac{\mathrm{f(x_1)}}{p}+\frac{\mathrm{f'(x_1)}}{1!}(t_1)\equiv 0\ (mod\ p)$ относительно t_1:

$$t_1 = t_1^{\nabla}(mod\ p)$$

Это, опять-таки, означает, что $t_1 = t_1^{\nabla} + pt_2$, где $t_2 \in$**Z**, и
$x = x_1 + pt_1 = x_1 + pt_1^{\nabla} + p^2t_2 = x_2 + p^2t_2$.
Снова вставим это $x = x_2 + p^2t_2$ в сравнение $f(x) \equiv 0(\mathrm{mod}\ p^3)$ (но теперь это сравнение уже по mod p^3), разложим его левую часть по формуле Тейлора по степеням $(x - x^2)$ и отбросим члены, кратные p^3:

$$\mathrm{f(x_2)}+\frac{\mathrm{f'(x_2)}}{1!}(p^2t_2)\equiv 0\ (mod\ p^3)$$

Делим это сравнение и его модуль на p^2:

$$\frac{\mathrm{f(x_2)}}{p^2}+\frac{\mathrm{f'(x_2)}}{1!}(t_2)\equiv 0\ (mod\ p)$$

Сравнения по составному модулю

Опять-таки $\frac{\mathrm{f(x_2)}}{p^2}$ – целое число, ведь число t_1^{∇} такое, что f $(\mathrm{x}_1 + \mathrm{p} \cdot\ t_1^{\nabla}) \equiv 0$ (mod p^2). Кроме того, $x_2 \equiv x_1(\mathrm{mod\ p})$, значит f '(x_2) ≡ f '(x_1)(mod p), т.е. f '(x_2) , как и f '(x_1) , не делится на p. Имеем единственное решение сравнения первой степени $\frac{\mathrm{f(x_2)}}{p^2} + \frac{\mathrm{f'(x_2)}}{1!}(t_2) \equiv 0\ (mod\ p)$ относительно t_2:

$$t_2 = t_2^{\nabla}(mod\ p)$$

Это, опять-таки, означает, что $t_2 = t_2^{\nabla} + pt_3$, где $t_3 \in$ **Z**, и

$$x = x_2 + p^2 t_2 = x_2 + p^2 t_2^{\nabla} + p^3 t_3 = \ x_3 + p^3 t_3$$

и процесс продолжается дальше и дальше, аналогично предыдущим шагам, до достижения степени α, в которой стоит простое число p в модуле исходного сравнения f (x) ≡ 0(mod p^{α}) .

Итак:

Всякое решение $x \equiv x_1$(mod p) ***сравнения*** f (x) ≡ 0(mod p) ***, при условии*** p/| f (x_1) ***, дает одно решение сравнения*** f (x) ≡ 0(mod p^{α}) ***вида*** $x \equiv x_{\alpha} + p^{\alpha} t_{\alpha}$***, т.е.*** $x \equiv x_{\alpha}$ (mod p^{α}) .♦

План лекции

Командная строка	**Решение СЛАУ**	**Вычеты, КТО**	**Группы, кольца, поля (начало)**
10 минут	**55 минут**	**20 минут**	**5 минут**

Группы, кольца, поля

НАЧАЛО!