А.П.Мостовской

Численные методы и система Mathematica

Учебное пособие

УДК 519.6(075.8)+004.42(075.8)
ББК 22.19я73+32.97я73
М 84

Автор: А.П.Мостовской.

Численные методы и система Mathematica:
Учебное пособие.- Мурманск: 2009.-249 с.

Пособие по учебной дисциплине Численные методы специальности 010.200 Прикладная математика и информатика. Пособие составлено в соответствии с государственным образовательным стандартом высшего профессионального образования по специальности 010.200 Прикладная математика и информатика.

Введение

Данное пособие предназначено преподавателям курса "Численные методы" и студентам, выполняющим практические задания или лабораторные работы по этому курсу. Пособие состоит их двух частей - курса лекций по "Численным методам" и примерам лабораторных работ по разделам этого курса, выполненным в широко известной системе компьютерных вычислений Mathematica[1].

Пособие может служить хорошей основой для проведения дистанционного обучения или подготовки студентов заочного отделения физико-математических факультетов по данному курсу.

В пособии приведены примеры численного решения задач в системе Mathematica по всем основным разделам линейной алгебры, аппроксимации и интерполяции функций, численного интегрирования, численного решения дифференциальных уравнений и некоторых задач численной оптимизации. Объем рассмотренного материала соответствует курсу "Численные методы" для студентов специальностей прикладная математика и информатика, математические методы в экономике. Пособие также будет полезно и студентам физико-математических факультетов педагогических университетов.

Для выполнения лабораторных работ можно было использовать любой из математических пакетов: Mathematica, Maple, MathLab, Gauss и другие, использовать любой язык программирования. Практика применения различных средств программирования при написании лабораторных работ показала, что наиболее гибкой, обладающей широким возможностями, позволяющими решить практически любые задачи, и не только задачи лабораторных работ по численным методам, является система Mathematica. Программирование в системе Mathematica позволяет просто и наглядно, без ненужного технического программирования, реализовать вычислительные схемы численных методов.

Система Mathematica была разработана фирмой Wolfram Research и постоянно совершенствуется. В данный момент выпущена версия 7, отличающаяся от предыдущих версий, в частности, большей скоростью вычислений. Команды, приведенные в данном пособии, соответствуют, в основном, версиям пакета 6 и 7.

Выполнение лабораторных работ в данном пакете требует знания основ и навыков программирования. На младших курсах, на таких факультетах, как физико-математический, в курсе "Информационные технологии в математике" или на вычислительной практике специальностей прикладная математика и информатика есть возможность подготовить студентов к основам работы в пакете Mathematica.

Примеры вычислительных схем, рассмотренные в данном пособии, позволяют лучшего понять соответствующий теоретический материал численных методов. С другой стороны, данное пособие можно рассматривать как практикум по программированию в системе Mathematica.

Ряд примеров пособия содержит три уровня программирования вычислительной задачи: построчное написание и выполнение команд, объединение команд в более компактную запись и построение команды, универсально решающей задачу. Для выполнения лабораторных работ достаточно построчного выполнения команд. Такой подход позволяет понять, как работает каждая отдельная команда, увидеть, что результат действия команды совпадает с ожидаемым. Для матриц больших размерностей,

[1] см. www.wolfram.com.

например, как в методе Гаусса, можно построчно выполнить несколько начальных команд для уяснения закономерности построения команд, и затем применить команды цикла для более компактной записи программы. Более компактное программирование позволяет применять метод в дальнейшем, при решении других вычислительных задач.

Каждая задача начинается с рассмотрения необходимого теоретического материала, вывода основных формул, на основе которых решается задача. Приведенные выводы и доказательства относятся к теоретической части курса и необходимы при построении алгоритмов численного решения задач.

Условия задач, содержащие матрицы, обычно генерируется с помощью команды *Random*[], так что элементы матрицы находятся в нужном диапазоне, матрица обладает нужными свойствами. Такой подход позволяет экономить время студентов при выполнении лабораторных заданий и задавать матрицы любого порядка. Далее идет разбор решения задачи вместе с приведением соответствующих команд системы Mathematica. Строчки текста, начинающиеся со служебного слова In[i]:= являются командами пакета Mathematica, где i - номер команды. Такие команды (без служебного слова) следует вводить в рабочем поле программы Mathematica в порядке их нумерации. Для выполнения команды нужно нажать клавиши Shift+Enter (или только клавишу правый Enter). Результат действия i-ой команды программа Mathematica выделяет в тексте служебным словом Out[i]= . Краткое описание каждой команды, встреченной в тексте, приведено в Приложении (см. также [6], [11], [12], [14], [15], [16]). Также строение команды можно посмотреть в Help'е программы. Для этого команду надо выделить курсором и нажать на клавишу F_1.

Наконец, заметим, что материал данного пособия может служить основой для учебно - исследовательской работы студентов, написания курсовых и дипломных работ.

Теория погрешностей

Решения практических задач методом математического моделирования обычно содержат погрешности, и поэтому полученные решения являются приближениями реальных процессов. Перечислим основные источники погрешностей.

1). **Погрешность исходных данных**. Этот вид погрешностей называется неустранимой погрешностью. Знание такой погрешности позволяет упростить решение самой задачи, а метод решения должен быть согласован с требуемой точностью. Точность вычислений по выбранному методу не должна превышать точность проведенных измерений. Вычислитель не имеет влияние на этот тип погрешностей. Конечно, вычислитель может потребовать перемерить исходные данные с большей точностью и, тем самым, возможно повысить точность результата.

Так, например, распределения тепла в плоской прямоугольной изотропной пластине в зависимости от времени есть одна из задач математической физике. Измерение температур в точках пластины в начальный момент времени есть приближенный процесс и возникшие при измерении погрешности относятся к погрешностям исходных данных.

2). **Погрешность метода решения.** Математическая модель практической задачи может быть сложной и не решаемой. Такая математическая задача подменяется

другой, более простой. Дифференциальное уравнение распределения температуры в пластине можно решить только численными методами. Применяя разностный метод, решение дифференциального уравнения сводится к решению системы линейных алгебраических уравнений, что создает новый вид погрешностей, которые относим к погрешностям метода.

3). **Погрешности округления.** Неизбежные округления, возникающие в процессе решения задачи, принадлежат к третьему типу погрешностей - погрешностям округления.

Рассмотрим традиционные вопросы теории погрешностей.

Абсолютная и относительная погрешности. Для двух чисел a и $a^\star$ разность

$$\Delta a = a^\star - a$$

назовем *абсолютной погрешностью* числа a. В этом равенстве отражена идеализированная практика измерений, в которой $a^\star$ есть точное и обычно недостижимое значение измеряемой величины, а a - это один из результатов измерения рассматриваемой величины. Так как $a^\star$ не известно, то неизвестна и абсолютная погрешность.

Величина Δ_a, взятая в разумных пределах и такая, что

$$\Delta_a \geqslant |\Delta a|$$

называется *предельной абсолютной погрешность.*

Отсюда получаем

$$|a^\star - a| \leqslant \Delta_a$$

или

$$a - \Delta_a \leqslant a^\star \leqslant a + \Delta_a \tag{1}$$

Таким образом, зная результат измерения и предельную абсолютную погрешность, можно найти границы, в которые заключено точное значение.

Обычно (1) записывают так

$$a^\star = a \pm \Delta_a$$

Пример. Линейкой измеряется длина стола. Допустим, что линейка такова, что при любом измерении a: a *см.* $- 0.05$ *см* $\leqslant a^\star \leqslant a$ *см.* $+ 0.05$ *см.*, где $a^\star$ - точное значение длины стола. Это значит, что $a^\star = a$ *см.* ± 0.05 *см.*, а предельная абсолютная погрешность таких измерений $\Delta_a = 0.05$ *см.* Если одно измерение показало, что $a = 184$ *см.*, то считаем, что длина стола равна 184 *см.* с погрешностью 0.05 *см.*

Предельная абсолютная погрешность не позволяет судить о том, хорошо или плохо произведено измерение. Например, измерение $l_1 = 28.4 \pm 0.1$ $(см)$ менее точное, чем $l_2 = 1128.3 \pm 0.1$ $(см)$.

Относительной погрешностью δa приближенного числа a называется отношение

$$\delta a = \frac{\Delta a}{|a|} \tag{2}$$

Произвольное число δ_a такое, что

$$\delta_a \geqslant |\delta a|$$

называется *предельной относительной погрешностью* приближенного числа a. Отсюда и из (2) получаем

$$|\Delta a| \leqslant |a|\delta_a$$

Следовательно $|a|\delta_a$ можно принять за предельную абсолютную погрешность

$$\Delta_a = |a|\delta_a \tag{3}$$

Обычно предельная относительная погрешность выражается в процентах. Так, предельная относительная погрешность длины стола в примере выше можно найти из формулы (3): $\delta_a = \frac{\Delta_a}{|a|} \cdot 100\% = \frac{0.05}{184} \cdot 100\% = 0.027\%$.

Пример. Какое из приближенных равенств точнее: $a_1^* = \frac{13}{19} \approx a_1 = 0.684$ или $a_2^* = \sqrt{52} \approx a_2 = 7.21$. При этом будем считать, что приближенное число a точнее приближенного числа b, если $\delta_a < \delta_b$.

Найдем предельные абсолютные погрешности этих чисел. Так как $|\Delta a_1| = |a_1^* - a_1| = |\frac{13}{19} - 0.684| = |0.68421... - 0.684| = 0.00021... \leqslant 0.00022 = \Delta_{a_1}$. Отсюда по формуле (3) получим, что $\delta_{a_1} = \frac{\Delta_{a_1}}{|a_1|} = \frac{0.00022}{0.684} = 0.00033 = 0.033\%$.

Аналогично для второго измерения получим, что $|\Delta a_2| \leqslant 0.0012 = \Delta_{a_2}$. Отсюда следует что $\delta_{a_2} = \frac{\Delta_{a_2}}{|a_2|} = \frac{0.0012}{7.21} = 0.00017 = 0.017\% < \delta_{a_1}$. Следовательно второе измерение точнее.

Значащие и верные цифры. Положительное число a можно записать в виде конечной иди бесконечной десятичной дроби:

$$a = \alpha_0 10^n + \alpha_1 10^{n-1} + ... + \alpha_k 10^{n-k} + ... \tag{4}$$

где α_k - цифры числа, $\alpha_0 \neq 0$, n - старший десятичный разряд, а цифра α_k стоит в разряде $n - k$. Например,

$$207.45 = 2 \cdot 10^2 + 0 \cdot 10^1 + 7 \cdot 10^0 + 4 \cdot 10^{-1} + 5 \cdot 10^{-2}.$$

Все цифры числа, начиная с первой левой ненулевой, называются *значащими цифрами*. Например, подчеркнутые цифры значащие: $\underline{24.344}$, $0.0\underline{34400}$.

Пусть значащая цифра α приближенного числа a, записанного в виде (4), стоит в k разряде, то есть

$$a = ... + \alpha_k 10^{n-k} + ...$$

Значащая цифра α называется *верной*, если модуль абсолютной погрешности числа меньше половины разряда, в котором стоит цифра:

$$|\Delta a| \leqslant 0.5 \cdot 10^{n-k}$$

Значащая цифра α называется *верной в широком смысле*, если модуль абсолютной погрешности числа меньше единицы разряда, в котором стоит цифра:

$$|\Delta a| \leqslant 10^{n-k}$$

Остальные цифры числа называются сомнительными цифрами.

В приближенном числе обычно оставляют только верные цифры. При этом, если последние нулевые цифры числа, стоящие после запятой, (например, 0.034400) не являются верными цифрами, то их не стоит писать.

Все цифры стоящие левее от верной цифры являются верными цифрами. Действительно, пусть цифра α_m стоит левее верной цифры α_k в приближенном числе (4), то есть $m < k$. Так как α_k верная цифра, то $|\Delta a| < 10^{n-k}$, а так как $10^{n-k} < 10^{n-m}$ при $m < k$, то отсюда получим, что $|\Delta a| < 10^{n-m}$. Следовательно цифра α_m является верной цифрой.

Пример. Округлить до верных знаков приближенное число $a = 25.384$, если абсолютная погрешность $\Delta a = 0.004$.

Так как $|\Delta a| > 0.5 \cdot 10^{-3} = 0.0005$, то цифра 4 в числе a не является верной цифрой. Цифра 8 является верной, так как $|\Delta a| < 0.5 \cdot 10^{-2} = 0.005$. Округляем a до сотых знаков: $a_1 = 25.38$. Проверим, будет ли a_1 состоять из верных цифр. Если число a есть приближенное значение для точного числа a^*, то $\Delta a = a^* - a = 0.004$, а погрешность округления $\Delta_{окр.} = a - a_1 = 25.384 - 25.38 = 0.004$. Тогда для приближенного числа a_1 (к числу a^*) абсолютная погрешность $\Delta a_1 = a^* - a_1 = a^* - a + a - a_1 = \Delta a + \Delta_{окр.} = 0.004 + 0.004 = 0.008$. Таким образом, есть точное число a^*, приближенное число $a_1 = 25.38$ и абсолютная погрешность $\Delta a_1 = 0.008$. Так как $|\Delta a_1| > 0.5 \cdot 10^{-2} = 0.005$, то цифра 8 не будет верной в приближенном числе a_1.

Начинаем все с начала - округляем приближенное число $a = 25.384$ до десятичных знаков, получим приближенное число $a_1 = 25.4$. Найдем абсолютную погрешность нового приближенного числа $\Delta a_1 = \Delta a + \Delta_{окр.} = 0.004 - 0.016 = -0.012$. Так как $|\Delta a_1| < 0.5 \cdot 10^{-1} = 0.05$, то цифра 4 будет верной в приближенном числе a_1. Таким образом, приближенное число a, округленное до верных знаков, есть число $a_1 = 25.4$.

Оценка погрешности результата вычислений.

Найдем оценку погрешности результата вычислений по формуле $u = u(x_1, ..., x_n)$, где $u(x_1, ..., x_n)$ - дифференцируемая функция.

Пусть $x_1, ..., x_n$ - приближенные значения величин $x_1^\star, ..., x_n^\star$ соответственно, u и $u^\star$ соответствующие значения функции. Допуская, что приращение функции приближенно можно заменить дифференциалом функции, запишем оценку абсолютной погрешности вычислений:

$$|\Delta u| = |u^\star - u| \approx |du| = \Big| \sum_{i=1}^{n} \frac{\partial u}{\partial x_i}(x_1, ..., x_n)(x_i^\star - x_i) \Big| \leqslant \sum_{i=1}^{n} \Big| \frac{\partial u}{\partial x_i}(x_1, ..., x_n) \Big| |\Delta x_i|.$$

Таким образом получаем следующую удобную оценку

$$|\Delta u| \leqslant \sum_{i=1}^{n} \Big| \frac{\partial u}{\partial x_i}(x_1, ..., x_n) \Big| |\Delta x_i|. \tag{4}$$

Следовательно правую часть можно принять за предельную абсолютную погрешность

$$\Delta_u = \sum_{i=1}^{n} \Big| \frac{\partial u}{\partial x_i} \Big| |\Delta x_i|$$

а если предварительно в (4) учесть неравенства $\Delta_{x_i} \geqslant |\Delta x_i|$, то получаем следующую формулу

$$\Delta_u = \sum_{i=1}^{n} \Big| \frac{\partial u}{\partial x_i} \Big| \Delta_{x_i} \tag{4'}$$

Найдем оценку предельной относительной погрешности. Из определения относительной погрешности и (4) получаем

$$|\delta u| = \frac{|\Delta u|}{|u|} \leqslant \sum_{i=1}^{n} \Big|\frac{1}{u}\frac{\partial u}{\partial x_i}\Big||\Delta x_i|$$

или

$$|\delta u| \leqslant \sum_{i=1}^{n} \Big|\frac{\partial \ln u}{\partial x_i}\Big||\Delta x_i|. \tag{5}$$

Следовательно правую часть можно принять за предельную относительную погрешность, а, если предварительно учесть оценку $\Delta_{x_i} \geqslant |\Delta x_i|$, то получим

$$\delta_u = \sum_{i=1}^{n} \Big|\frac{\partial \ln u}{\partial x_i}\Big|\Delta_{x_i}. \tag{6}$$

Следствия. Докажем справедливость следующих равенств. В равенствах 1 и 2 предполагается, что числа x_1 и x_2 одного знака.

1. $\Delta_{x_1+x_2} = \Delta_{x_1} + \Delta_{x_2}, \;\; \delta_{x_1+x_2} = \delta_{x_1} + \delta_{x_2}.$
2. $\Delta_{x_1-x_2} = \Delta_{x_1} + \Delta_{x_2}, \;\; \delta_{x_1-x_2} = \dfrac{|x_1|\delta_{x_1} + |x_2|\delta_{x_2}}{|x_1 - x_2|}.$
3. $\Delta_{x_1\cdot x_2} = |x_1 \cdot x_2|\delta_{x_1\cdot x_2}$, где $\delta_{x_1\cdot x_2} = \delta_{x_1} + \delta_{x_2}.$
4. $\Delta_{\frac{x_1}{x_2}} = |\dfrac{x_1}{x_2}|\delta_{\frac{x_1}{x_2}}$, где $\delta_{\frac{x_1}{x_2}} = \delta_{x_1} + \delta_{x_2}.$
5. $\delta_{\sqrt[m]{x}} = \dfrac{1}{m}\delta_x.$

Доказательство. 1. Пусть $u(x_1, x_2) = x_1 + x_2$. Так как $\dfrac{\partial u}{\partial x_i} = 1, \;\; i = 1, 2$, то по формуле (4′) получаем $\Delta_{x_1+x_2} = \Delta_{x_1} + \Delta_{x_2}$.

Применяя к такой функции формулу (5), получим

$$|\delta(x_1 + x_2)| \leqslant \frac{1}{|x_1 + x_2|}|\Delta x_1| + \frac{1}{|x_1 + x_2|}|\Delta x_2| \leqslant \frac{1}{|x_1|}\Delta_{x_1} + \frac{1}{|x_2|}\Delta_{x_2}$$

так как x_1 и x_2 одного знака и $|\Delta x_1| \leqslant \Delta_{x_1}$, $|\Delta x_2| \leqslant \Delta_{x_2}$. Отсюда и из (3) получаем

$$|\delta(x_1 + x_2)| \leqslant \delta_{x_1} + \delta_{x_2}.$$

Правую часть можно принять за предельную относительную погрешность

$$\delta_{x_1+x_2} = \delta_{x_1} + \delta_{x_2}.$$

2. Если $u(x_1, x_2) = x_1 - x_2$, то, применяя формулу (4′), можно получить равенство $\Delta_{x_1-x_2} = \Delta_{x_1} + \Delta_{x_2}$. Отсюда и из (3) получим

$$|\delta(x_1 - x_2)| = \frac{|\Delta(x_1 - x_2)|}{|x_1 - x_2|} \leqslant \frac{\Delta_{x_1-x_2}}{|x_1 - x_2|} = \frac{\Delta_{x_1} + \Delta_{x_2}}{|x_1 - x_2|} = \frac{|x_1|\delta_{x_1} + |x_2|\delta_{x_2}}{|x_1 - x_2|}.$$

Следовательно

$$\delta_{x_1-x_2} = \frac{|x_1|\delta_{x_1} + |x_2|\delta_{x_2}}{|x_1 - x_2|}.$$

3. Пусть теперь $u(x_1, x_2) = x_1 \cdot x_2$. Подставим эту функцию в (5), получим

$$|\delta(x_1 \cdot x_2)| \leqslant \frac{|x_2||\Delta x_1|}{|x_1 \cdot x_2|} + \frac{|x_1||\Delta x_2|}{|x_1 \cdot x_2|} = \frac{|\Delta x_1|}{|x_1|} + \frac{|\Delta x_2|}{|x_2|} = |\delta x_1| + |\delta x_2| \leqslant \delta_{x_1} + \delta_{x_2}.$$

Поэтому можно положить

$$\delta_{x_1 \cdot x_2} = \delta_{x_1} + \delta_{x_2}.$$

Для вычисления $\Delta_{x_1 \cdot x_2}$ используем формулу (3).

4. Доказательство аналогично п.3.

5. Пусть $u = \sqrt[m]{x}$. Тогда $x = u^m$. Из п.3: $\delta_x = \delta_u + ... + \delta_u = m\delta_u$, что доказывает утверждения п.5.

Примеры. 1. Измерения прямоугольника показали, что точные значения его сторон равны $a^* = 5 \pm 0.5$ (см.), $b^* = 3 \pm 0.5$ (см.) Требуется найти точное значение его площади $S^* = S \pm \Delta_S$.

Приближенные значения сторон равны $a = 5$, $b = 3$, а $\Delta_a = \Delta_b = 0.5$. Поэтому приближенное значение площади $S = a \cdot b = 15$. Найдем δ_S. По формуле (3) $\delta_a = \frac{0.5}{5} = 0.1$, $\delta_b = \frac{0.5}{3} \approx 0.17$. По п.3 следствия $\delta_S = \delta_a + \delta_b = 0.1 + 0.17 = 0.27$. Отсюда и из (3) получаем $\Delta_S = |S| \cdot \delta_s = 15 \cdot 0.27 = 4.05$. Поэтому $S^* = 15 \pm 4.05$ (см^2.).

2. Все цифры приближенных чисел $a = 0.23$, $b = 23.45$, $c = 2.03$ верные. Найти точное значение $u = ln(a) \cdot b \cdot \sqrt{c}$.

Воспользуемся программой Mathematica. Вводим функцию

In[1]:= **u = Log[a] b Sqrt[c]**

Out[1]= b Sqrt[c] log(a)

Найдем предельную абсолютную погрешность по формуле (4). При этом, далее в командах вместо Δ_m будем писать Δm и понимать под новым обозначением предельную абсолютную погрешность. По формуле (6) запишем, что

In[2]:= **Δu = Abs[D[Log[u], a]] Δa + Abs[D[Log[u], b]] Δb + Abs[D[Log[u], a]] Δc**

Out[2]= $\frac{\Delta b}{|b|} + \left|\frac{1}{a\ \log(a)}\right|\Delta a + \frac{\Delta c}{2|c|}$

Так как все цифры данных приближенных чисел верные, то можно определить предельную абсолютную погрешность каждого числа. Так в числе a последняя цифра 3 - верная, значит $|\Delta a| \leqslant 0.5 \cdot 10^{-2} = 0.005$. Значит, можно положить $\Delta_a = 0.005$. Аналогично получаем $\Delta_b = \Delta_c = 0.005$.

Вводим предельные абсолютные погрешности и сами числа:

In[3]:= **Δa = Δb = Δc = 0.005;**

In[4]:= **a = 0.23; b = 23.45; c = 2.03;**

Теперь можно вычислить предельную абсолютную погрешность значения функции. Следующая команда вызывает выражение для предельной абсолютной погрешности значения функции, в которое подставляет все значения параметров.

In[5]:= **Δu**

Out[5]= 0.0162

Находим значение функции на данных значениях аргумента a, b, c.

In[6]:= **u**

Out[6]= -49.1035

Теперь точное значение $u^* = u \pm \Delta_u = -49.1035 \pm 0.0162$.

Так как предельные погрешности положительны, то при больших вычисления накапливается слишком большие погрешности. Поэтому приведенные формулы в таких случаях не применяются. Приведем практические правила академика Крылова, которыми можно руководствоваться при больших вычислениях.

При умножении и делении приближенных чисел, вообще говоря, с различным числом значащих цифр производится округление результата с числом значащих цифр, совпадающим с минимальным числом значащих цифр у исходных чисел.

При сложении или вычитании результат округляется по минимальному числу верных цифр после запятой у исходных чисел.

Глава 1

Численные методы линейной алгебры

В этой главе рассматриваются численные методы решений систем алгебраический уравнений, вычисление определителей, обратной матрицы и нахождение собственных значений и векторов квадратных матриц.

Рассмотрим систему линейных алгебраических уравнений

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n = b_2, \\ \quad . \quad . \quad . \quad . \\ a_{k1}x_1 + a_{k2}x_2 + ... + a_{kn}x_n = b_k, \end{cases} \qquad (*)$$

с матрицей

$$A = \begin{pmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ . & . & . & . \\ a_{k1} & a_{k2} & ... & a_{kn} \end{pmatrix} \qquad (**)$$

и столбцом свободных членов $b = (b_1, b_2, ..., b_k)^\top$.

Пусть $x = (x_1, x_2, ..., x_n)^\top$ - неизвестные системы. Тогда систему (*) можно записать в матричном виде

$$A.x = b.$$

Все численные методы решения алгебраических задач делятся на прямые (точные) методы и итерационные методы. Прямые методы приводят к решению за конечное число арифметических операций, а если при этом не делать округлений, то решение задачи получится точным. Итерационные методы позволяют находить решение приближенно, в результате построения достаточного числа элементов последовательности, сходящейся к решению.

К прямым методам можно отнести метод Крамера, по которому решение системы $(*)$ при $k = n$ можно найти из равенств

$$x_i = \frac{\det A_i}{\det A}, \quad i = 1, 2, ..., n,$$

где матрица A_i получается из матрицы системы A заменой i-го столбца матрицы A на столбец свободных членов b. Если, при вычислении определителя затрачивать $n!$ арифметических операций, то, например, при $n = 100$ для получения решения системы методом Крамера надо затратить не меньше $100! = 10^{158}$ операций. Такое число операций не доступно для вычислений.

Аналогично обстоит дело с решением системы $(*)$ при $k = n$ с помощью обратной матрицы

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

Вычисление обратной матрицы также связано с вычислением определителей. Так как на практике встречаются системы уравнений с очень большим числом неизвестных, то приведенные методы, обычно, не применяются на практике.

Норма вектора, матрицы. Нормой элемента x линейного векторного пространства называется вещественное число $||x||$ такое, что

1) $||x|| \geqslant 0$; $||x|| = 0$ тогда и только тогда, когда $x = 0$.
2) $||\lambda x|| = |\lambda| \cdot ||x||$ для всех чисел λ.
3) $||x + y|| \leqslant ||x|| + ||y||$.

Для вектора $x = (x_1, x_2, ..., x_n)$ рассматриваются следующие нормы:

$$||x||_1 = \sum_{i=1}^{n} |x_i|, \quad ||x||_\infty = \max_i |x_i|,$$

и евклидова норма

$$||x||_2 = \sqrt{\sum_{i=1}^{n} x_i^2}.$$

Для матрицы $A = (a_{ij})$ порядка $n \times m$ рассматриваются нормы

$$||A||_1 = \max_{i=1,...,n} \sum_{j=1}^{m} |a_{ij}|, \quad ||A||_\infty = \max_{j=1,...,m} \sum_{i=1}^{n} |a_{ij}|$$

и евклидова норма матрицы

$$||A|| = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij}^2}.$$

§1 Системы линейных уравнений. Определители. Обращение матриц

В данном параграфе рассматриваются прямые методы решения систем линейных алгебраических уравнений, основанные на методе Гаусса: прямой метод Гаусса (метод исключения неизвестных) и метод Гаусса с выбором главного элемента, вычисление определителя матрицы с применением метода Гаусса, метод Жордана обращения матриц.

n.1 Метод Гаусса решения систем линейных уравнений

Рассмотрим систему линейных алгебраических уравнений относительно n неизвестных $x_1, ..., x_n$ с матрицей системы $A = (a_{ij})$ порядка n и столбцом свободных членов $b = (b_1, b_2, ..., b_k)^\top$. Требуется найти решение $x = (x_1, ..., x_n)$ такой системы уравнений методом Гаусса. Будем считать, что матрица системы невырождена.

Прямой ход метода Гаусса. Пусть коэффициент a_{11} не равен нулю (в противном случае придется переставить местами соответствующие уравнения системы). Поделим первое уравнение системы (ведущая строка) на элемент a_{11} (ведущий элемент):

$$\begin{cases} x_1 + a_{12}^{(1)} x_2 + ... + a_{1n}^{(1)} x_n = b_1^{(1)}, \\ a_{21} x_1 + a_{22} x_2 + ... + a_{2n} x_n = b_2, \\ \quad . \quad . \quad . \quad . \\ a_{n1} x_1 + a_{n2} x_2 + ... + a_{nn} x_n = b_n, \end{cases} \tag{1}$$

Здесь и далее индексами вида "(1)" обозначены соответствующие изменения коэффициентов. Исключим из такой системы переменную x_1 из второго, третьего и так далее до n-го уравнения. Для этого из i-го уравнения системы (1), $i = 2, 3, ..., n$, вычитаем первое уравнение, умноженное на коэффициент a_{i1}. В результате приведем систему к следующему виду.

$$\begin{cases} x_1 + a_{12}^{(1)} x_2 + ... + a_{1n}^{(1)} x_n = b_1^{(1)}, \\ \quad a_{22}^{(1)} x_2 + ... + a_{2n}^{(1)} x_n = b_2^{(1)}, \\ \quad . \quad . \quad . \quad . \\ \quad a_{n2}^{(1)} x_2 + ... + a_{nn}^{(1)} x_n = b_n^{(1)}. \end{cases}$$

Далее, делим второе уравнение на коэффициент при x_2 и исключаем переменную x_2 из третьего и так далее до n-го уравнения. Повторяем этот процесс до последнего уравнения. Последнее уравнение поделим на коэффициент при неизвестной. Тем самым, равносильными преобразованиями, система уравнений будет приведена к треугольному виду.

$$\begin{cases} x_1 + a_{12}^{(1)} x_2 + a_{13}^{(1)} x_3 + ... + a_{1,n-1}^{(1)} x_{n-1} + a_{1n}^{(1)} x_n = b_1^{(1)}, \\ \quad x_2 + a_{23}^{(2)} x_3 + ... + a_{2,n-1}^{(2)} x_{n-1} + a_{2n}^{(2)} x_n = b_2^{(2)}, \\ \quad . \quad . \quad . \quad . \\ \quad x_{n-1} + a_{n-1,n}^{(n-1)} x_n = b_{n-1}^{(n-1)}, \\ \quad x_n = b_n^{(n)}, \end{cases} \tag{2}$$

Отметим, что на практике такому преобразованию подвергают не уравнения системы, а строки расширенной матрицы системы и приводят расширенную матрицу исходной системы к расширенной матрице системы (2).

Обратный ход метода Гаусса. Из последнего уравнения системы (2), содержащего одну переменную, находим x_n. Найденное x_n подставляем в предпоследнее уравнение и находим x_{n-1}, и так далее. В общем виде, формулы для определения неизвестных имеют следующий вид:

$$x_n = b_n^{(n)}, \quad x_k = b_k^{(k)} - \sum_{j=k+1}^{n} a_{kj}^{(k)} x_j, \quad k = n-1, ..., 1.$$

В этом заключается обратный ход метода Гаусса.

Невязка решения. По методу Гаусса решение системы получено за конечное число арифметических операций, поэтому метод Гаусса есть точный метод. Неизбежные округления дают определенную погрешность, величину которой характеризует так называемый вектор невязки

$$\xi = b - A.x,$$

где A - матрица исходной системы, а x найденный вектор решения. Любая норма вектора невязки называется невязкой решения.

Число арифметических операций в методе Гаусса. При исключении неизвестной x_1 производится n делений второго и следующих коэффициентов первого уравнения и его правой части на коэффициент при x_1. Далее, совершаем n умножений и n вычитаний при исключении x_1 из 2-го, 3-го и последующих уравнений, число которых равно $n-1$. Таким образом, число операций при исключении x_1 равно $Q_1 = n + 2n(n-1) = 2n^2 - n = 2(n-1+1)^2 - (n-1+1)$.

При исключении x_2 производятся аналогичные вычисления, число которых на 1 меньше, то есть $Q_2 = n-1+2(n-1)(n-1-1) = 2(n-2+1)^2-(n-2+1)$. И так далее, при исключении x_i число арифметических операций равно $Q_i = 2(n-i+1)^2-(n-i+1)$. При $i = n$, $Q_n = 1$, что соответствует приведению последнего уравнения системы к виду, приведенному в (2), то есть к виду, последнее уравнение которого поделено на коэффициент при x_n.

Всего арифметических операций при прямом ходе метода Гаусса

$$Q_p = \sum_{i=1}^{n} Q_i = \sum_{i=1}^{n} [2(n-i+1)^2 - (n-i+1)].$$

Совершим в данной сумме замену $k = n - i + 1$, воспользуемся двумя известными формулами

$$\sum_{i=1}^{n} k = \frac{n+1}{2} n, \qquad \sum_{i=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6},$$

и окончательно получим

$$Q_p = \sum_{k=n}^{1} (2k^2 - k) = \sum_{k=1}^{n} (2k^2 - k) = 2\sum_{k=1}^{n} k^2 - \sum_{k=1}^{n} k =$$

$$2\frac{n(n+1)(2n+1)}{6} - \frac{n+1}{2} n = \frac{2}{3}n^3 + \frac{n^2}{2} - \frac{n}{6} \approx \frac{2}{3}n^3.$$

В обратном ходе число арифметических операций в два раза больше, чем число элементов под главной диагональю матрицы системы, то есть

$$Q_o = n^2 - n.$$

Суммируем число операций прямого и обратного ходов, получим

$$Q = Q_p + Q_o = \frac{2}{3}n^3 + \frac{n^2}{2} - \frac{n}{6} + n^2 - n = \frac{2}{3}n^3 + \frac{3n^2}{2} - \frac{7n}{6} \approx \frac{2}{3}n^3.$$

Таким образом, решение системы n уравнений с n неизвестными методом Гаусса требует примерно $\frac{2n^3}{3}$ арифметических операций.

Отметим, что число арифметических операций над правой частью системы в прямом ходе метода Гаусса при исключении x_1 равно $q_1 = 1 + 2(n-1) = 2n-1$, при исключении x_2 равно $q_2 = 1+2(n-2) = 2(n-2+1)-1$ и так далее, при исключении x_i равно $q_i = 1+2(n-i) = 2(n-i+1)-1$. Отсюда получаем общее число операций:

$$q = \sum_{i=1}^{n}[2(n-i+1)-1] = \sum_{k=1}^{n}(2k-1) = n(n+1)-n = n^2.$$

Теперь можно оценить число арифметических операций при решении методом Гаусса p систем, отличающихся только правой частью. Такие системы не надо решать отдельно, а решить надо одну систему с "расширенной правой частью", содержащей правые части всех систем (см. п.3). При этом будет проведено следующее число арифметических операций:

$$Q(p) = \frac{2}{3}n^3 + \frac{3n^2}{2} - \frac{7n}{6} + p(n^2+n^2-n) \approx \frac{2}{3}n^3 + 2pn^2.$$

Рассмотрим пример. Дана расширенная матрица

In[1]:= $\mathbf{m} = \{\{\mathbf{2},\mathbf{4},\mathbf{7},\mathbf{2},\mathbf{-3},\mathbf{5}\},\{\mathbf{-1},\mathbf{3},\mathbf{4},\mathbf{1},\mathbf{3},\mathbf{2}\},\{\mathbf{4},\mathbf{3},\mathbf{-5},\mathbf{2},\mathbf{3},\mathbf{1}\},$
$\{\mathbf{1},\mathbf{5},\mathbf{2},\mathbf{-2},\mathbf{-3},\mathbf{-8}\},\{\mathbf{4},\mathbf{-3},\mathbf{-1},\mathbf{8},\mathbf{4},\mathbf{2}\}\};$

системы уравнений

$$\begin{cases} 2x_1 + 4x_2 + 7x_3 + 2x_4 - 3x_5 = 5 \\ -x_1 + 3x_2 + 4x_3 + x_4 + 3x_5 = 2 \\ 4x_1 + 3x_2 - 5x_3 + 2x_4 + 3x_5 = 1 \\ x_1 + 5x_2 + 2x_3 - 2x_4 - 3x_5 = -8 \\ 4x_1 - 3x_2 - x_3 + 8x_4 + 4x_5 = 2 \end{cases}$$

Решить данную систему методом Гаусса.

Проверим правильность введения матрицы[1]

In[2]:= $\mathbf{m}//\mathbf{MatrixForm}$

Out[2]:=
$$\begin{pmatrix} 2 & 4 & 7 & 2 & -3 & 5 \\ -1 & 3 & 4 & 1 & 3 & 2 \\ 4 & 3 & -5 & 2 & 3 & 1 \\ 1 & 5 & 2 & -2 & -3 & -8 \\ 4 & -3 & -1 & 8 & 4 & 2 \end{pmatrix}$$

Обозначим через

In[3]:= $\mathbf{b} = \mathbf{m}[[\mathbf{All},\mathbf{6}]]$

[1] В Mathematica 5, если включить опцию Cell→ DefaultOutputFormatType→ TraditionalForm, то матрица выводится в привычном виде сразу или при вводе только команды m, а команда m // MatrixForm не обязательна. В Mathematica 6,7 для этого нужно открыть форму Edit → Preferences и на вкладке Evaluation в строке Format type of new output cells выбрать TraditionalForm.

Out[3]= {5,2,1,-8,2}

столбец свободных членов системы уравнений и введем матрицу системы, удаляя из матрицы m последний столбец[1]:

In[4]:= $\mathbf{a = m[[1;;,1;;5]]}$;

Out[4]=

$$\begin{pmatrix} 2, & 4 & 7 & 2 & -3 \\ -1 & 3 & 4 & 1 & 3 \\ 4 & 3 & -5 & 2 & 3 \\ 1 & 5 & 2 & -2 & -3 \\ 4 & -3 & -1 & 8 & 4 \end{pmatrix}$$

Прямой ход. Перед каждым шагом исключения нужно проверять, не является ли элемент главной диагонали ведущей строки нулевым, чтобы не получить деления на ноль.

Коэффициент (ведущий элемент) $m[[1,1]] = 2 \neq 0$ не равен нулю. Делим первое уравнение системы на $m[[1,1]]$ и исключаем переменную x_1 из второго, третьего и так далее до 5-го уравнения[2]:

In[5]:=
$\mathbf{m[[1]] = m[[1]]/m[[1,1]]//N;}$
$\mathbf{m[[2]] = m[[2]] - m[[1]]\ m[[2,1]]//N;}$
$\mathbf{m[[3]] = m[[3]] - m[[1]]\ m[[3,1]]//N;}$
$\mathbf{m[[4]] = m[[4]] - m[[1]]\ m[[4,1]]//N;}$
$\mathbf{m[[5]] = m[[5]] - m[[1]]\ m[[5,1]]//N;}$

Проверим результат выполнения операций:

Int[6]:= $\mathbf{m//MatrixForm}$

Out[6]=

$$\begin{pmatrix} 1. & 2. & 3.5 & 1. & -1.5 & 2.5 \\ 0. & 5. & 7.5 & 2. & 1.5 & 4.5 \\ 0. & -5. & -19. & -2. & 9. & -9. \\ 0. & 3. & -1.5 & -3. & -1.5 & -10.5 \\ 0. & -11. & -15. & 4. & 10. & -8. \end{pmatrix}$$

Исключаем переменную $\mathbf{x_2}$ из третьего и так далее до 5-го уравнения:

In[7]:=
$\mathbf{m[[2]] = m[[2]]/m[[2,2]]//N;}$
$\mathbf{m[[3]] = m[[3]] - m[[2]]\ m[[3,2]]//N;}$
$\mathbf{m[[4]] = m[[4]] - m[[2]]\ m[[4,2]]///N;}$
$\mathbf{m[[5]] = m[[5]] - m[[2]]\ m[[5,2]]//N;}$

Проверим результат выполнения операций:

Int[8]:= $\mathbf{m//MatrixForm}$

Out[8]=

$$\begin{pmatrix} 1. & 2. & 3.5 & 1. & -1.5 & 2.5 \\ 0 & 1. & 1.5 & 0.4 & 0.3 & 0.9 \\ 0 & 0 & -11.5 & 0 & 10.5 & -4.5 \\ 0 & 0 & -6. & -4.2 & -2.4 & -13.2 \\ 0 & 0 & 1.5 & 8.4 & 13.3 & 1.9 \end{pmatrix}$$

---

[1] В Mathematica 5, здесь и везде далее, вместо команды $a = m[[1;;,1;;5]]$; применяется команда $(a = Transpose[Drop[Transpose[m], -1]])//MatrixForm$

Отметим, что данная команда, введенная без так расставленных круглых скобок, не приведет к определению матрицы a.

[2] Здесь и далее, приведенная нумерация команд несколько отличается от нумерации программы.

Исключаем переменную $\mathbf{x_3}$ из 4-го и 5-го уравнений:

In[9]:=
```
m[[3]] = m[[3]]/m[[3, 3]]//N;
m[[4]] = m[[4]] - m[[3]] m[[4, 3]]//N;
m[[5]] = m[[5]] - m[[3]] m[[5, 3]]//N;
```

Проверим результат выполнения операций:

In[10]:=
```
m//MatrixForm
```

Out[10]=

$$\begin{pmatrix} 1. & 2. & 3.5 & 1. & -1.5 & 2.5 \\ 0 & 1. & 1.5 & 0.4 & 0.3 & 0.9 \\ 0 & 0 & 1. & 0 & -0.913043 & 0.391304 \\ 0 & 0 & 0 & -4.2 & -7.87826 & -10.8522 \\ 0 & 0 & 0 & 8.4 & 14.6696 & 1.31304 \end{pmatrix}$$

Исключаем переменную $\mathbf{x_4}$ из 5-го уравнения:

In[11]:=
```
m[[4]] = m[[4]]/m[[4, 4]]//N;
m[[5]] = m[[5]] - m[[4]] m[[5, 4]]//N;
```

Проверим результат выполнения операций:

In[12]:=
```
m//MatrixForm
```

Out[12]=

$$\begin{pmatrix} 1. & 2. & 3.5 & 1. & -1.5 & 2.5 \\ 0 & 1. & 1.5 & 0.4 & 0.3 & 0.9 \\ 0 & 0 & 1. & 0 & -0.913043 & 0.391304 \\ 0 & 0 & 0 & 1. & 1.87578 & 2.58385 \\ 0 & 0 & 0 & 0 & -1.08696 & -20.3913 \end{pmatrix}$$

Наконец, делим последнюю строку на коэффициент $m[[5, 5]]$.

In[13]:=
```
m[[5]] = m[[5]]/m[[5, 5]]//N;
m//MatrixForm
```

Out[13]=

$$\begin{pmatrix} 1. & 2. & 3.5 & 1. & -1.5 & 2.5 \\ 0 & 1. & 1.5 & 0.4 & 0.3 & 0.9 \\ 0 & 0 & 1. & 0 & -0.913043 & 0.391304 \\ 0 & 0 & 0 & 1. & 1.87578 & 2.58385 \\ 0 & 0 & 0 & 0 & 1. & 18.76 \end{pmatrix}$$

Таким образом, расширенная матрица системы уравнений приведена к треугольному виду, а система уравнений приняла следующий вид:

$$\begin{cases} x_1 + 2x_2 + 3.5x_3 + x_4 - 1.5x_5 = 2.5 \\ x_2 + 1.5x_3 + 0.4x_4 + 0.3x_5 = 0.9 \\ x_3 - 0.913043x_5 = 0.391304 \\ x_4 + 1.87578x_5 = 2.58385 \\ x_5 = 18.76 \end{cases}$$

Обратный ход. Из последнего уравнения системы находим

In[14]:=
```
x5 = m[[5, 6]]
```

Out[14]= 18.76

из предпоследнего уравнения находим

In[15]:=
```
x4 = m[[4, 6]] - m[[4, 5]]x5
```

Out[15]= -32.6057

из 3-го уравнения находим

In[16]:= $\mathbf{x_3 = m[[3,6]] - m[[3,5]]x_5 - m[[3,4]]x_4}$

Out[16]= 17.52

из 2-го уравнения находим

In[17]:= $\mathbf{x_2 = m[[2,6]] - m[[2,5]]x_5 - m[[2,4]]x_4 - m[[2,3]]x_3}$

Out[17]= -17.9657

наконец, из 1-го уравнения находим

In[18]:= $\mathbf{x_1 = m[[1,6]] - m[[1,5]]x_5 - m[[1,4]]x_4 - m[[1,3]]x_3 - m[[1,2]]x_2}$

Out[18]= 37.8571

Найдем вектор невязки $r = b - a.x$ полученного решения. Введем вектор решения

In[19]:= $\mathbf{x = \{x_1, x_2, x_3, x_4, x_5\}}$

Out[19]= {37.8571, -17.9657, 17.52, -32.6057, 18.76}

и вычислим вектор невязки

In[20]:= $\mathbf{r = b - a.x}$

Out[20]= {0,0,0,0,0}

Невязка $\sqrt{r.r}$ равна нулю, следовательно, все команды, приводящие к решению системы уравнений, введены верно[1].

Приведем к более компактному виду программу. Выполним снова команду 1, то есть, введем матрицу m, и применим команду цикла для объединения команд прямого хода:

In[2]:=
```
Do[m[[j]] = m[[j]]/m[[j,j]]//N;
   Do[m[[i]] = m[[i]] - m[[j]]m[[i,j]]//N, {i, 2 + j - 1, 5}],
 {j, 1, 5}]
```

Здесь учтена следующая особенность программы Mathematica - цикл, у которого начальное значение итерации больше конечного: $Do[expr, \{i, 6, 5\}]$, не выполняются. Поэтому, при $j = 5$, произойдет только деление последней строки на коэффициент $m[[5, 5]]$, а внутренний цикл не выполнится.

Команда Table (можно применить и команду Do) позволяет компактно записать обратный ход метода Гаусса:

In[3]:= $\mathbf{Table[x_i = m[[i,6]] - \sum_{j=i+1}^{5} m[[i,j]]]x_j, \{i, 5, 1, -1\}]}$

Out[3]= {18.76,-32.6057,17.52,-17.9657,37.8571}

Отметим, что сумма, у которой нижняя граница суммирования больше верхней, равна нулю. Поэтому, при $i = 5$ в $Table$ будет только присваивание вида $x_5 = m[[5, 6]]$.

Выведем ответ:

In[4]:= $\mathbf{x = \{x_1, x_2, x_3, x_4, x_5\}}$

Out[4]= {37.8571, -17.9657, 17.52, -32.6057, 18.76}

Введем теперь команду, реализующую метод Гаусса для произвольной расширенной матрицы системы линейных алгебраических уравнений.

Сначала введем команду swp перестановки строк матрицы. Аргумент команды можно дополнить командой $m\_?MatrixQ$, которая тестирует данный аргумент - является ли он матрицей. Если введенный аргумент не матрица, то значение команды swp

[1] См. конец §2.

с таким аргументом не вычисляется. Аналогично проверяется аргумент командами вида $i\_?IntegerQ$.

```
In[1]:= swp[m_?MatrixQ, i_?IntegerQ, j_?IntegerQ] :=
          Block[{p = m}, p[[i]] = m[[j]]; p[[j]] = m[[i]]; p]
```

Следующая команда $gauss[n]$ содержит прямой и обратный ход и дополнена проверкой ведущего элемента - является ли он нулем? Если на некотором шаге прямого хода ведущий элемент $m[[j, j]]$ равен нулю, то в качестве ведущего элемента выбирается последний в данном j-ом столбце нижележащий элемент отличный от нуля. Такой элемент всегда существует в силу невырожденности матрицы системы. Для этого командой

$$s = Take[m[[All, j]], \{j + 1, st\}]$$

из j-го столбца выбираются все элементы, расположенные ниже ведущего элемента $m[[j, j]]$ (нулевого). Из этого списка элементов выбирается один ненулевой элемент, например последний:

$$e = Last[Select[s, (\# \neq 0)\&]]$$

Далее, выбираем позицию найденного элемента в j столбце:

$$k = Last[Position[m[[All, j]], e]//Flatten];$$

Здесь команда $Flatten$ убирает лишние скобки, а команда $Last$ выбирает последнюю позицию элемента e в j столбце из, возможно, нескольких вариантов.

Наконец, командой $m = swp[m, j, k]$ меняем местами j и k строки и прямой ход продолжается.

```
In[2]:= gauss[n_?MatrixQ] := Block[{m = n, st, sb, a, b, s, e, k, r, x},
(* Размерность расширенной матрицы, столбец свободных членов, матрица системы *)
          {st, sb} = Dimensions[m];
          b = m[[All, sb]];
          a = m[[1;;, 1;;st]];
          If[Det[a] == 0,
          Print["Определитель системы равен нулю"]; Abort[]];
(* Прямой ход *)
          Do[
(* Перестановка строк при нулевом ведущем элементе *)
            If[m[[j, j]] == 0,
              s = Take[m[[All, j]], {j + 1, st}];
              e = Last[Select[s, (\# \neq 0)\&]];
              k = Last[Position[m[[All, j]], e]//Flatten];
              m = swp[m, j, k]
             ];
(* Обнуление поддиагональных элементов *)
            m[[j]] = m[[j]]/m[[j, j]]//N;
            Do[m[[i]] = m[[i]] - m[[j]]m[[i, j]]//N, {i, 2 + j - 1, st}],
          {j, 1, st}];
```

(* Обратный ход *)

$$\mathbf{Table}[\mathbf{x_i} = \mathbf{m}[[\mathbf{i},\mathbf{sb}]] - \sum_{\mathbf{j=i+1}}^{\mathbf{st}} \mathbf{m}[[\mathbf{i},\mathbf{j}]]\mathbf{x_j}, \{\mathbf{i},\mathbf{st},\mathbf{1},-\mathbf{1}\}];$$

(* Векторы решения и невязки *)

$$\mathbf{x} = .; \mathbf{x} = \mathbf{Table}[\mathbf{x_i}, \{\mathbf{i},\mathbf{1},\mathbf{st}\}];$$

$$\mathbf{r} = \mathbf{b} - \mathbf{a}.\mathbf{x}//\mathbf{Chop};$$

$$\mathbf{Print}[\text{"}\mathbf{Решение\ \ x} = \text{"}, \mathbf{x}, \text{"}\mathbf{c\ вектором\ невязки\ \ r} = \text{"}, \mathbf{r}, \text{"."}]]$$

Перед применением команды $gauss[m]$ нужно ввести расширенную матрицу системы, например, повторно ввести матрицу **m** и выполнить команду

In[3]:= **gauss[m]**

Out[3]= Решение x={37.8571,-17.9657,17.52,-32.6057,18.76} с вектором невязки r={0,0,0,0,0}.

n.2 Метод Гаусса с выбором главного элемента

Метод Гаусса приводит к большим погрешностям решения при округлении, если ведущие элементы метода исключения близки к нулю. В этом случае применяют метод Гаусса с выбором главного элемента. Этот метод отличается от обычного метода Гаусса только выбором ведущей строки и ведущего элемента на каждом шаге прямого хода. В качестве ведущего элемента берем максимальный по модулю элемент среди элементов матрицы системы (не расширенной матрицы!). Строка, содержащая ведущий элемент является ведущей. Применяя схему Гаусса к расширенной матрице системы, обнуляем все коэффициенты системы, стоящие над и под ведущим элементом. Затем мысленно вычеркиваем ведущую строку из расширенной матрицы и с оставшимися элементами расширенной матрицы системы поступаем точно также. При этом, элементы мысленно вычеркнутой строки не обнуляем. Повторяем этот процесс до последнего уравнения и получаем систему с треугольной матрицей, но с переставленными строками и, возможно, столбцами.

Обратный ход аналогичен обратному ходу прямого метода Гаусса - первое неизвестное находим из уравнения с одним ненулевым коэффициентом и свободным членом, второе из уравнения с двумя ненулевыми коэффициентами и так далее.

Рассмотрим пример. Пусть дана система с расширенной матрицей

In[1]:= $\mathbf{m} = \{\{\mathbf{2},\mathbf{4},\mathbf{7},\mathbf{2},-\mathbf{3},\mathbf{5}\},\{-\mathbf{1},\mathbf{3},\mathbf{4},\mathbf{1},\mathbf{3},\mathbf{2}\},\{\mathbf{4},\mathbf{3},-\mathbf{5},\mathbf{2},\mathbf{3},\mathbf{1}\},$
$\{\mathbf{1},\mathbf{5},\mathbf{2},-\mathbf{2},-\mathbf{3},-\mathbf{8}\},\{\mathbf{4},-\mathbf{3},-\mathbf{1},\mathbf{8},\mathbf{4},\mathbf{2}\}\}$

Проверим правильность введения матрицы

In[2]:= **m//MatrixForm**

Out[2]=

$$\begin{pmatrix} 2 & 4 & 7 & 2 & -3 & 5 \\ -1 & 3 & 4 & 1 & 3 & 2 \\ 4 & 3 & -5 & 2 & 3 & 1 \\ 1 & 5 & 2 & -2 & -3 & -8 \\ 4 & -3 & -1 & 8 & 4 & 2 \end{pmatrix}$$

Видим, что наибольшим по модулю элементом матрицы системы является 8 в позиции (5,4). Обнуляем все коэффициенты, стоящие над и под этим ведущим элементом. Но, сначала делим ведущую, 5 строку на ведущий элемент. Тем самым делаем коэффициент при x_4 в ведущей строке системы равным единице.

In[3]:=
$$\mathbf{m[[5]] = m[[5]]/m[[5,4]]//N;}$$
$$\mathbf{m[[1]] = m[[1]] - m[[5]]m[[1,4]]//N;}$$
$$\mathbf{m[[2]] = m[[2]] - m[[5]]m[[2,4]]//N;}$$
$$\mathbf{m[[3]] = m[[3]] - m[[5]]m[[3,4]]//N;}$$
$$\mathbf{m[[4]] = m[[4]] - m[[5]]m[[4,4]]//N;}$$

Проверим результат выполнения операций:

In[4]:= **m//MatrixForm**

Out[4]=
$$\begin{pmatrix} 1. & 4.75 & 7.25 & 0. & -4. & 4.5 \\ -1.5 & 3.375 & 4.125 & 0. & 2.5 & 1.75 \\ 3. & 3.75 & -4.75 & 0. & 2. & 0.5 \\ 2. & 4.25 & 1.75 & 0. & -2. & -7.5 \\ 0.5 & -0.375 & -0.125 & 1. & 0.5 & 0.25 \end{pmatrix}$$

Видим, что среди элементов новой матрицы системы, максимальным по модулю элементом является элемент в позиции (1,3). Обнуляем все коэффициенты, стоящие над и под этим новым ведущим элементом, кроме коэффициента предыдущей ведущей строки.

In[5]:=
$$\mathbf{m[[1]] = m[[1]]/m[[1,3]]//N;}$$
$$\mathbf{m[[2]] = m[[2]] - m[[1]]m[[2,3]]//N;}$$
$$\mathbf{m[[3]] = m[[3]] - m[[1]]m[[3,3]]//N;}$$
$$\mathbf{m[[4]] = m[[4]] - m[[1]]m[[4,3]]//N;}$$

Проверим результат выполнение операций:

In[6]:= **m//MatrixForm//Chop**

Out[6]=
$$\begin{pmatrix} 0.137931 & 0.655172 & 1. & 0. & -0.551724 & 0.62069 \\ -2.06897 & 0.672414 & 0. & 0. & 4.77586 & -0.810345 \\ 3.65517 & 6.86207 & 0. & 0. & -0.62069 & 3.44828 \\ 1.75862 & 3.10345 & 0. & 0. & -1.03448 & -8.58621 \\ 0.5 & -0.375 & -0.125 & 1. & 0.5 & 0.25 \end{pmatrix}$$

Мысленно вычеркиваем 1 и 5 предыдущие ведущие строки и среди оставшихся элементов матрицы системы находим максимальный по модулю элемент. Это элемент в позиции (3,2). Обнуляем все коэффициенты системы, стоящие над и под этим элементом, кроме элементов мысленно вычеркнутых строк.

In[7]:=
$$\mathbf{m[[3]] = m[[3]]/m[[3,2]]//N;}$$
$$\mathbf{m[[2]] = m[[2]] - m[[3]]m[[2,2]]//N;}$$
$$\mathbf{m[[4]] = m[[4]] - m[[3]]m[[4,2]]//N;}$$

Проверим результат выполнения операций:

In[8]:= **m//Chop//MatrixForm**
$$\begin{pmatrix} 0.137931 & 0.655172 & 1. & 0 & -0.551724 & 0.62069 \\ -2.42714 & 0 & 0 & 0 & 4.83668 & -1.14824 \\ 0.532663 & 1. & 0 & 0 & -0.0904523 & 0.502513 \\ 0.105528 & 0 & 0 & 0 & -0.753769 & -10.1457 \\ 0.5 & -0.375 & -0.125 & 1. & 0.5 & 0.25 \end{pmatrix}$$

Мысленно вычеркиваем 1, 3 и 5 строки, а в оставшихся строках матрицы системы находим максимальный по модулю элемент. Это элемент в позиции (2,5). Обнуляем коэффициенты в четвертом столбце под и над элементом (2,5) но не обнуляем элементы предыдущих ведущих строк.

In[9]:= $\mathbf{m[[2]] = m[[2]]/m[[2,5]]//N;}$
$\mathbf{m[[4]] = m[[4]] - m[[2]]m[[4,5]]//N;}$

In[10]:= $\mathbf{m//MatrixForm}$

Out[10]=
$$\begin{pmatrix} 0.137931 & 0.655172 & 1. & 0. & -0.551724 & 0.62069 \\ -0.501818 & 0. & 0. & 0. & 1. & -0.237403 \\ 0.532663 & 1. & 0. & 0. & -0.0904523 & 0.502513 \\ -0.272727 & 0 & 0. & 0. & 0. & -10.3247 \\ 0.5 & -0.375 & -0.125 & 1. & 0.5 & 0.25 \end{pmatrix}$$

Наконец, выполним команду

In[11]:= $\mathbf{m[[4]] = m[[4]]/m[[4,1]]//N;}$

и приведем расширенную матрицу системы к виду

In[12]:= $\mathbf{m//MatrixForm//Chop}$

Out[12]=
$$\begin{pmatrix} 0.137931 & 0.655172 & 1. & 0 & -0.551724 & 0.62069 \\ -0.501818 & 0 & 0 & 0 & 1. & -0.237403 \\ 0.532663 & 1. & 0 & 0 & -0.0904523 & 0.502513 \\ 1. & 0 & 0 & 0 & 0 & 37.8571 \\ 0.5 & -0.375 & -0.125 & 1. & 0.5 & 0.25 \end{pmatrix}$$

Если поменять местами соответствующие строки и столбцы этой матрицы, то получим треугольную расширенную матрицу системы. Но, это делать не обязательно.

Таким образом, исходная система равносильными преобразованиями приведена к виду:

$$\begin{cases} 0.137931x_1 + 0.655172x_2 + x_3 - 0.551724x_5 = 0.62069, \\ -0.501818x_1 + x_5 = -0.237403, \\ 0.532663x_1 + x_2 - 0.0904523x_5 = 0.502513, \\ x_1 = 37.8571, \\ 0.5x_1 - 0.375x_2 - 0.125x_3 + x_4 + 0.5x_5 = 0.25. \end{cases}$$

Обратный ход такой же как и в прямом методе Гаусса. Выбираем строку матрицы системы с двумя коэффициентами. Это 4 строка. Из нее находим

In[13]:= $\mathbf{x_1 = m[[4,6]]}$

Out[13]= 37.8571

Из второго уравнения системы находим

In[14]:= $\mathbf{x_5 = m[[2,6]] - m[[2,1]]x_1}$

Out[14]= 18.76

Из третьего уравнения системы находим

In[15]:= $\mathbf{x_2 = m[[3,6]] - m[[3,1]]x_1 - m[[3,5]]x_5}$

Out[15]= -17.9657

Из первого уравнения системы находим

In[16]:= $\mathbf{x_3 = m[[1,6]] - m[[1,1]]x_1 - m[[1,2]]x_2 - m[[1,5]]x_5}$

Out[16]= 17.52

и, наконец, из пятого уравнения системы находим

In[17]:= $\mathbf{x_4 = m[[5,6]] - m[[5,1]]x_1 - m[[5,2]]x_2 - m[[5,3]]x_3 - m[[5,5]]x_5}$

Out[17]= -32.6057

Найдем невязку полученного решения. Введем вектор решения

In[18]:= $\mathbf{x = \{x_1, x_2, x_3, x_4, x_5\}}$

Out[18]= {37.8571, -17.9657, 17.52, -32.6057, 18.76}

и вычислим вектор невязки

In[19]:= $\mathbf{r = b - a.x//Chop}$

Out[19]= {0,0,0,0,0}

Невязка равна нулю, следовательно решение системы уравнений найдено верно.

Введем элементы программирования в решение предыдущей задачи. Они помогут составить программу метода Гаусса с выбором главного элемента.

Начнем все с начала. Пусть дана система с расширенной матрицей

In[1]:= $\mathbf{m = \{\{2, 4, 7, 2, -3, 5\}, \{-1, 3, 4, 1, 3, 2\}, \{4, 3, -5, 2, 3, 1\},}$
$\mathbf{\{1, 5, 2, -2, -3, -8\}, \{4, -3, -1, 8, 4, 2\}\}}$

Введем матрицы: столбец свободных членов системы уравнений и матрицу системы.

In[2]:= $\mathbf{b = m[[All, 6]];}$

In[3]:= $\mathbf{a = a1 = m[[1;;, 1;;5]];}$

Введем дополнительную переменную, в ней будут содержаться номера ведущих строк.

In[4]:= $\mathbf{dr = \{\};}$

Прямой ход. 1 шаг.

Наибольший по модулю коэффициент матрицы системы определяется командой $Max[Abs[a1]]$, а его позиция в матрице системы, с элементами взятыми по абсолютной величине, командой

In[5]:= $\mathbf{p = Position[Abs[a1], Max[Abs[a1]]][[1]]}$

Out[5]= {5,4}

Максимальный по модулю элемент матрицы системы расположен в 5 строке и 4 столбце.

In[6]:= $\mathbf{dr = Append[dr, First[p]]}$

Out[6]= {5}

Таким образом, ведущим максимальным по модулю элементом является $m[[5,4]] = 8$, а список dr содержит номер ведущей строки - 5. Исключаем переменную x_4 из всех уравнений, кроме 5-го:

In[7]:=
$$\mathbf{m[[5]] = m[[5]]/m[[5, 4]]//N;}$$
$$\mathbf{m[[1]] = m[[1]] - m[[5]]m[[1, 4]]//N;}$$
$$\mathbf{m[[2]] = m[[2]] - m[[5]]m[[2, 4]]//N;}$$
$$\mathbf{m[[3]] = m[[3]] - m[[5]]m[[3, 4]]//N;}$$
$$\mathbf{m[[4]] = m[[4]] - m[[5]]m[[4, 4]]//N;}$$

Проверим результат выполнения операций:

Int[9]:= $\mathbf{m//MatrixForm}$

Out[9]=
$$\begin{pmatrix} 1 & 4.75 & 7.25 & 0 & -4 & 4.5 \\ -1.5 & 3.375 & 4.125 & 0 & 2.5 & 1.75 \\ 3 & 3.75 & -4.75 & 0 & 2 & 0.5 \\ 2 & 4.25 & 1.75 & 0 & -2 & -7.5 \\ 0.5 & -0.375 & -0.125 & 1 & 0.5 & 0.25 \end{pmatrix}$$

2 шаг.

Возьмем копию матрицы полученной системы, она будет играть вспомогательную роль.

In[10]:= **a1 = m[[1;;,1;;5]];**

и заменим в ней элементы ведущей строки нулями, чтобы они не участвовали в выборе второго ведущего элемента:

In[11]:= **a1[[dr]] = Table[0,{5}];**

Int[12]:= **a1//MatrixForm**

Out[12]=

$$\begin{pmatrix} 1 & 4.75 & 7.25 & 0 & -4 \\ -1.5 & 3.375 & 4.125 & 0 & 2.5 \\ 3 & 3.75 & -4.75 & 0 & 2 \\ 2 & 4.25 & 1.75 & 0 & -2 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Теперь находим второй максимальный по модулю элемент среди элементов матрицы $Abs[a1]$.

In[13]:= **p = Position[Abs[a1], Max[Abs[a1]]][[1]]**

Out[13]= {1,3}

Список ведущих строк:

In[14]:= **dr = Append[dr, First[p]]**

Out[14]= {5,1}

Обнуляем все коэффициенты матрицы m над и под элементом $m[[1,3]] = 7.25$, но не обнуляем элемент предыдущей ведущей строки.

In[15]:=
```
m[[1]] = m[[1]]/m[[1, 3]]//N;
m[[2]] = m[[2]] - m[[1]]m[[2, 3]]//N;
m[[3]] = m[[3]] - m[[1]]m[[3, 3]]//N;
m[[4]] = m[[4]] - m[[1]]m[[4, 3]]//N;
```

Проверим результат выполнение операций:

Int[16]:= **m//MatrixForm//Chop**

Out[16]=

$$\begin{pmatrix} 0.137931 & 0.655172 & 1. & 0. & -0.551724 & 0.62069 \\ -2.06897 & 0.672414 & 0. & 0. & 4.77586 & -0.810345 \\ 3.65517 & 6.86207 & 0. & 0. & -0.62069 & 3.44828 \\ 1.75862 & 3.10345 & 0. & 0. & -1.03448 & -8.58621 \\ 0.5 & -0.375 & -0.125 & 1. & 0.5 & 0.25 \end{pmatrix}$$

3 шаг.

Повторяем предыдущий шаг.

Возьмем копию матрицы полученной системы

In[17]:= **a1 = m[[1;;,1;;5]];**

и заменим в ней элементы предыдущих ведущих строк нулями. Номера ведущих строк содержаться в списке dr, поэтому результат команды

In[18]:= **a1[[dr]] = Table[0,{5}];**

будет следующая матрица:

Int[19]:= **a1//MatrixForm**

Out[19]=

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ -2.07 & 0.67 & 0 & 0 & 4.78 \\ 3.66 & 6.87 & 0 & 0 & -0.62 \\ 1.76 & 3.10 & 0 & 0 & -1.03 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Теперь находим третий максимальный по модулю элемент

In[20]:= **p = Position[Abs[a1], Max[Abs[a1]]][[1]]**

Out[20]= {3,2}

Составим список ведущих строк:

In[21]:= **dr = Append[dr, First[p]]**

Out[21]= {5,1,3}

Обнуляем элементы над и под ведущим элементом матрицы m в строках, не включенных в список dr :

In[22]:= **m[[3]] = m[[3]]/m[[3, 2]]//N;**
m[[2]] = m[[2]] − m[[3]]m[[2, 2]]//N;
m[[4]] = m[[4]] − m[[3]]m[[4, 2]]//N;

Проверим результат выполнения операций:

Int[23]:= **m//Chop//MatrixForm**

Out[23]=

$$\begin{pmatrix} 0.137931 & 0.655172 & 1. & 0 & -0.551724 & 0.62069 \\ -2.42714 & 0 & 0 & 0 & 4.83668 & -1.14824 \\ 0.532663 & 1. & 0 & 0 & -0.0904523 & 0.502513 \\ 0.105528 & 0 & 0 & 0 & -0.753769 & -10.1457 \\ 0.5 & -0.375 & -0.125 & 1. & 0.5 & 0.25 \end{pmatrix}$$

4 шаг.

Возьмем копию матрицы полученной системы

In[24]:= **a1 = m[[1; ; , 1; ; 5]];**

элементы предыдущих ведущих строк заменим нулями

In[25]:= **a1[[dr]] = Table[0, {5}];**

Int[26]:= **a1//MatrixForm**

Out[26]=

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ -2.42714 & 0 & 0 & 0 & 4.83668 \\ 0 & 0 & 0 & 0 & 0 \\ 0.105528 & 0 & 0 & 0 & -0.753769 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Находим максимальный по модулю элемент среди элементов матрицы $Abs[a1]$:

In[27]:= **p = Position[Abs[a1], Max[Abs[a1]]][[1]]**

Out[27]= {2,5}

In[28]:= **dr = Append[dr, First[p]]**

Out[28]= {5,1,3,2}

Обнуляем коэффициенты в четвертой строке под элементом $m[[2, 5]] = 4.78$, но не обнуляем элементы предыдущих ведущих строк.

In[29]:= **m[[2]] = m[[2]]/m[[2, 5]]//N;**
m[[4]] = m[[4]] − m[[2]]m[[4, 5]]//N;

Int[30]:= **m//Chop//MatrixForm**

$$\begin{pmatrix} 0.137931 & 0.655172 & 1. & 0. & -0.551724 & 0.62069 \\ -0.501818 & 0. & 0. & 0. & 1. & -0.237403 \\ 0.532663 & 1. & 0. & 0. & -0.0904523 & 0.502513 \\ -0.272727 & 0 & 0. & 0. & 0. & -10.3247 \\ 0.5 & -0.375 & -0.125 & 1. & 0.5 & 0.25 \end{pmatrix}$$

Наконец, выполним команду

In[31]:= $\mathbf{m[[4]] = m[[4]]/m[[4,1]]//N;}$

и приведем расширенную матрицу системы к виду

In[31]:= $\mathbf{m//Chop//MatrixForm}$

Out[31]=

$$\begin{pmatrix} 0.137931 & 0.655172 & 1. & 0 & -0.551724 & 0.62069 \\ -0.501818 & 0 & 0 & 0 & 1. & -0.237403 \\ 0.532663 & 1. & 0 & 0 & -0.0904523 & 0.502513 \\ 1. & 0 & 0 & 0 & 0 & 37.8571 \\ 0.5 & -0.375 & -0.125 & 1. & 0.5 & 0.25 \end{pmatrix}$$

Обратный ход уже был проведен выше.

Проведем сокращение программы. Опять начнем все с начала. Введем расширенную матрицу системы.

In[1]:= $\mathbf{m = \{\{2,4,7,2,-3,5\},\{-1,3,4,1,3,2\},\{4,3,-5,2,3,1\},}$
$\mathbf{\{1,5,2,-2,-3,-8\},\{4,-3,-1,8,4,2\}\};}$

Проверим правильность ввода

In[2]:= $\mathbf{m//MatrixForm}$

Число уравнений

In[3]:= $\mathbf{n = Length[m];}$

Столбец свободных членов

In[4]:= $\mathbf{b = m[[All, n+1]];}$

Введем пустые списки, в которым будут добавляться номера строки и столбца, содержащих ведущий элемент на каждом этапе:

In[5]:= $\mathbf{dr = \{\};}$
$\mathbf{dc = \{\};}$

и вспомогательный список, состоящий из номеров строк системы:

In[6]:= $\mathbf{tb = Table[i, \{i, 1, n\}]}$

Out[6]= {1,2,3,4,5}

Матрица системы и, равная ей на первом шаге, вспомогательная матрица $a1$:

In[7]:= $\mathbf{a1 = a = m[[1;;, 1;;n]];}$

В следующем цикле объединены все шаги прямого хода метода Гаусса с выбором главного элемента. Цикл проработает n раз. Команда

$$k = Complement[tb, dr]$$

убирает из списка tb элементы списка dr, содержащего номера ведущих строк, полученных на данном шаге и, следовательно, список k содержит номера только тех строк, в которых будут обнуляться коэффициенты на данном шаге. Команда

$$(m[[\#]] = m[[\#]] - m[[p1]]m[[\#, p2]])\&/@k$$

подставляет вместо # очередной элемент списка k, тем самым выполняются команды обнуления элементов нужных строк расширенной матрицы системы на данном шаге.

In[8]:=
```
Do[
    a1 = m[[1;;,1;;n]];
    a1[[dr]] = Table[0,{n}];
(* Позиция максимального по модулю элемента в матрице системы *)
    p = Position[Abs[a1],Max[Abs[a1]]][[1]];
(* Номер строки ведущего элемента *)
    p1 = First[p];
(*Номер столбца ведущего элемента *)
    p2 = Last[p];
(* Список номеров строк ведущих элементов *)
    dr = Append[dr,p[[1]]];
(* Список номеров столбцов ведущих элементов *)
    dc = Append[dc,p[[2]]];
(* Список номеров строк, в которых будут обнуляться элементы на данном шаге
*)
    k = Complement[tb,dr];
(* Деление ведущей строки на ведущий элемент *)
    m[[p1]] = m[[p1]]/m[[p1,p2]]//N;
(* Обнуление элементов в строках с номерами в списке k над и под ведущем
элементом*)
    (m[[#]] = m[[#]] - m[[p1]]m[[#,p2]])&/@k//N;,
  {n}]
```

In[9]:= **m//MatrixForm**

Out[9]=

$$\begin{pmatrix} 0.137931 & 0.655172 & 1. & 0 & -0.551724 & 0.62069 \\ -0.501818 & 0 & 0 & 0 & 1. & -0.237403 \\ 0.532663 & 1. & 0 & 0 & -0.0904523 & 0.502513 \\ 1. & 0 & 0 & 0 & 0 & 37.8571 \\ 0.5 & -0.375 & -0.125 & 1. & 0.5 & 0.25 \end{pmatrix}$$

Для выполнения обратного хода, перепишем списки *dr*, *dc* следующим образом:

In[10]:= **dc = Reverse[dc]**

Out[10]= {1,5,2,3,4}

In[11]:= **dr = Reverse[dr]**

Out[11]= {4,2,3,1,5}

Теперь обратный ход можно записать так:

In[12]:= **$x_{dc[[1]]}$ = m[[dr[[1]],n+1]]/m[[dr[[1]],dc[[1]]]]//N**

Out[12]= 37.8571

In[13]:=
```
x_dc[[2]] = (m[[dr[[2]],n+1]]-
  m[[dr[[2]],dc[[2-1]]]]x_dc[[1]])/m[[dr[[2]],dc[[2]]]]//N
```

Out[13]= 18.76

In[14]:=
```
x_dc[[3]] = (m[[dr[[3]],n+1]] - m[[dr[[3]],dc[[1]]]]x_dc[[1]]-
  m[[dr[[3]],dc[[2]]]]x_dc[[2]])/m[[dr[[3]],dc[[3]]]]//N
```

Out[14]= -17.9657

In[15]:=
```
x_dc[[4]] = (m[[dr[[4]],n+1]]-
  m[[dr[[4]],dc[[1]]]]x_dc[[1]] - m[[dr[[4]],dc[[2]]]]x_dc[[2]]-
  m[[dr[[4]],dc[[3]]]]x_dc[[3]])/m[[dr[[4]],dc[[4]]]]//N
```

Out[15]= 17.52

```
In[16]:= x[[dc[[5]]]] = (m[[dr[[5]], n + 1]] -
          m[[dr[[5]], dc[[1]]]]x[[dc[[1]]]] - m[[dr[[5]], dc[[2]]]]x[[dc[[2]]]] -
          m[[dr[[5]], dc[[3]]]]x[[dc[[3]]]] - m[[dr[[5]], dc[[4]]]]x[[dc[[4]]]])/
          m[[dr[[5]], dc[[5]]]]//N
Out[16]= -32.6057
```

Объединим предыдущие команды обратного хода с 12 по 16 с помощью команды цикла

```
In[17]:= Do[
           x[[dc[[i]]]] = (m[[dr[[i]], n + 1]] - Sum[m[[dr[[i]], dc[[j]]]]x[[dc[[j]]]], {j, 1, i - 1}])//N,
         {i, 1, n}]
In[18]:= {x1, x2, x3, x4, x5}
Out[184]= {37.8571,-17.9657,17.52,-32.6057,18.76}
```

Создадим теперь команду, решающую систему линейных уравнений произвольного порядка методом Гаусса с выбором главного элемента.

```
In[1]:= gaussvge[m1_] :=
          Block[{m = m1, b, n = Length[m1], dr = {}, dc = {}},
           b = m[[All, n + 1]];
           tb = Table[i, {i, 1, n}];
           a1 = a = m[[1;; , 1;; n]];
           Do[
            a1 = m[[1;; , 1;; n]];
            a1[[dr]] = Table[0, {n}];
            p = Position[Abs[a1], Max[Abs[a1]]][[1]];
            p1 = First[p];
            p2 = Last[p];
            dr = Append[dr, p[[1]]];
            dc = Append[dc, p[[2]]];
            k = Complement[tb, dr];
            m[[p1]] = m[[p1]]/m[[p1, p2]]//N;
            (m[[#]] = m[[#]] - m[[p1]]m[[#, p2]])&/@k//N;,
           {n}];
           dc = Reverse[dc];
           dr = Reverse[dr];
           Do[
            x[[dc[[i]]]] = (m[[dr[[i]], n + 1]] - Sum[m[[dr[[i]], dc[[j]]]] x[[dc[[j]]]], {j, 1, i - 1}])//N,
           {i, 1, n}];
          Print["Решение x = ", xot = Table[xi, {i, 1, n}],
           "с вектором невязки r = ", b - a.xot//Chop]
          ]
```

Для применения команды, нужно ввести расширенную матрицу m и выполнить команду

```
In[2]:= gaussvge[m]
```

n.3 Вычисление обратной матрицы методом Жордана

Найдем обратную матрицу для невырожденной квадратной матрицы $m = (a_{ij})$ n-го порядка. Если x - обратная матрица, то

$$m.x = E, \tag{3}$$

где E - единичная матрица. Пусть $x_i = (x_{1i}, ..., x_{ni})^\top$ - i-ый столбец обратной матрицы x, а $e_i = (0, ..., 1, ..., 0)^\top$ - i-ый столбец матрицы E, $i = 1, ..., n$. Тогда одно матричное уравнение (3) равносильно n матричным уравнениям

$$m.x_1 = e_1, \;\; m.x_2 = e_2, \;\; \dots \;\; , m.x_n = e_n. \tag{4}$$

Системы (4) имеют одну и ту же матрицу системы, поэтому эти системы можно решить все вместе. Для этого присоединим к матрицы m справа единичную матрицу:

$$\begin{pmatrix} a_{11} & a_{12} & ... & a_{1n} & 1 & 0 & . & 0 \\ a_{21} & a_{22} & ... & a_{2n} & 0 & 1 & . & 0 \\ . & . & . & . & . & . & . & . \\ a_{n1} & a_{n2} & ... & a_{nn} & 0 & 0 & . & 1 \end{pmatrix}$$

и получим "объединенную" расширенную матрицу, то есть матрицу, содершащую расширенные матрицы всех систем. Применяя схему Гаусса, приведем полученную матрицу к виду

$$\begin{pmatrix} 1 & 0 & ... & 0 & b_{11} & b_{12} & . & b_{1n} \\ 0 & 1 & ... & 0 & b_{12} & b_{21} & . & b_{1n} \\ . & . & . & . & . & . & . & . \\ 0 & 0 & ... & 1 & b_{n1} & b_{n2} & . & b_{nn} \end{pmatrix} \tag{5}$$

Тем самым системы (4) равносильными преобразованиями будут приведены к виду

$$\begin{cases} x_{11} = b_{11}, \\ x_{21} = b_{21}, \\ \dots \\ x_{n1} = b_{n1}, \end{cases} , \quad \begin{cases} x_{12} = b_{12}, \\ x_{22} = b_{22}, \\ \dots \\ x_{n2} = b_{n2}, \end{cases} , \dots , \begin{cases} x_{1n} = b_{1n}, \\ x_{2n} = b_{2n}, \\ \dots \\ x_{nn} = b_{nn}, \end{cases} .$$

Решения каждой из этих систем есть соответствующий столбец обратной матрицы. Поэтому уберем из матрицы (5) столбцы единичной матрицу слева, получим обратную матрицу для матрицы m.

Из п.1 следует, что вычисление обратной матрицы методом Жордана требует примерно $3n^3$ арифметических операций.

Запрограммируем этот метод для частного случая. Пусть дана матрица m и ее копия a:

In[1]:=
```
m = a = {{2, 4, 7, 2, -3}, {-1, 3, 4, 1, 3}, {4, 3, -5, 2, 3},
         {1, 5, 2, -2, -3}, {4, -3, -1, 8, 4}}
```

Введем ее размерность и присоединим к матрице **m** справа единичную матрицу:

In[2]:=
```
n = Length[m];
m = Transpose[Join[Transpose[m], IdentityMatrix[n]]]
```

Out[2]=

$$\begin{pmatrix} 2 & 4 & 7 & 2 & -3 & 1 & 0 & 0 & 0 & 0 \\ -1 & 3 & 4 & 1 & 3 & 0 & 1 & 0 & 0 & 0 \\ 4 & 3 & -5 & 2 & 3 & 0 & 0 & 1 & 0 & 0 \\ 1 & 5 & 2 & -2 & -3 & 0 & 0 & 0 & 1 & 0 \\ 4 & -3 & -1 & 8 & 4 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Применяя процедуру исключения Гаусса к дополненной матрице **m**

In[3]:=
```
Do[
 Do[
  If[i != j, m[[i]] = m[[i]] - m[[j]]m[[i, j]]/m[[j, j]]//N],
 {i, 1, n}];
 m[[j]] = m[[j]]/m[[j, j]],
{j, 1, n}]
```

приведем матрицу **m** к виду

In[4]:=
```
m//Chop
```

Out[4]=

$$\begin{pmatrix} 1. & 0 & 0 & 0 & 0 & 2.3333 & -0.5714 & 1.7142 & -3.6666 & -1.8571 \\ 0 & 1. & 0 & 0 & 0 & -1.0533 & 0.2971 & -0.6914 & 1.7866 & 0.8457 \\ 0 & 0 & 1. & 0 & 0 & 1.0799 & -0.160 & 0.6799 & -1.6799 & -0.8399 \\ 0 & 0 & 0 & 1. & 0 & -1.9466 & 0.4171 & -1.4514 & 3.2133 & 1.7257 \\ 0 & 0 & 0 & 0 & 1. & 1.0399 & -0.080 & 0.8399 & -1.8399 & -0.9199 \end{pmatrix}$$

Вырезаем из матрицы m столбцы единичной матрицы слева, получим обратную матрицу t :

In[5]:=
```
t = m[[1;; , n + 1;; 2n]]
```

Out[5]=

$$\begin{pmatrix} 2.3333 & -0.5714 & 1.7142 & -3.6666 & -1.8571 \\ -1.0533 & 0.2971 & -0.6914 & 1.7866 & 0.8457 \\ 1.0799 & -0.160 & 0.6799 & -1.6799 & -0.8399 \\ -1.9466 & 0.4171 & -1.4514 & 3.2133 & 1.7257 \\ 1.0399 & -0.080 & 0.8399 & -1.8399 & -0.9199 \end{pmatrix}$$

Проверка

In[6]:=
```
a.t//Chop
```

Результатом проверки является единичная матрица.

Объединим все приведенные команды и напишем команд с именем, например, $jordan[m1]$, обращающую данную матрицу.

In[1]:=
```
jordan[m1_] := Block[{m = m1, n = Length[m1]},
  m = Transpose[Join[Transpose[m], IdentityMatrix[n]]];
  Do[
   m[[j]] = m[[j]]/m[[j, j]];
   Do[
     If[i != j, m[[i]] = m[[i]] - m[[j]] m[[i, j]]//N],
   {i, 1, n}];
  {j, 1, n}];
  m[[All, n + 1;; 2n]]]
```

Применение команды. Введем матрицу m повторно (команда In[1] предыдущего блока) и выполним команду

In[2]:=
```
jordan[m]
```

Результат действия команды уже приведен для данной матрицы (см. Out(5) предыдущего блока).

Проведем последнее улучшение программы. В предыдущей версии программы предполагалось, что все ведущие (диагональные) элементы матриц были отличны от нуля, а матрицы невырождены. Дополним предыдущую программу командами и для этого случая. Дополнительно предусмотрим вывод промежуточных результатов.

Введем команду перестановки строк матрицы:

```
In[1]:= swp[m_, i_, j_] :=
          Block[{p = m}, p[[i]] = m[[j]]; p[[j]] = m[[i]]; p]
```

Следующая команда $jordan[w, opt]$ содержит два аргумента - матрицу m и опцию opt, значением которой может быть y или любой другой символ. Если значение opt будет y, то программа выведет промежуточные результаты вычисления.

Команда $Select[m[[All, i]], (\#\ !\ = 0)\&]$ выбирает ненулевой элемент i-го столбца, а команда $Position$ возвращает номер строки k, в которой находится выбранный элемент, если элемент $m[[i, i]]$ равен нулю, и меняет эту k-ую строку с i-ой строкой.

```
In[2]:= jordan[w_, opt_] :=
        Block[{m = w, n = Length[w]},
         If[Det[m] == 0,
           Print["Нет обратной матрицы"]; Abort[]];
          m = Transpose[Join[Transpose[m], IdentityMatrix[n]]];
(* Перестановка строк матрицы (см. команду gauss) *)
          Do[
           If[m[[i, i]] == 0,
           s = Take[m[[All, i]], {i + 1, n}];
           e = Last[Select[s, (# ≠ 0)&]];
           k = Last[Position[m[[All, i]], e]//Flatten];
           m = swp[m, i, k]
          ];
             m[[i]] = m[[i]]/m[[i, i]];
(* Список f содержит номера строк, в которых будут обнуляться элемнетны на
данном шаге. Это номера всех строк, кроме i-ой. *)
              f = Complement[Range[n], {i}];
              (m[[#]] = m[[#]] - m[[i]] m[[#, i]])&/@f;
              If[ToString[opt] == "y", Print[m//MatrixForm]],
          {i, 1, n}];
          m[[1;; , n + 1;; 2n]]//N]
```

Применение команды. Введем произвольную квадратную матрицу m и выполним команду

```
In[3]:= jordan[m, y]
```

если требуются промежуточные вычисления или команду

```
In[4]:= jordan[m, n]
```

приводящую только ответ - обратную матрицу m^{-1}.

n.4 Вычисление определителей

Вычислим определитель квадратной матрицы

In[1]:= $\mathbf{m = m1 = Table[Random[], \{i, 1, 5\}, \{j, 1, 5\}]}$;

Out[1]=
$$\begin{pmatrix} 0.317823 & 0.53827 & 0.827947 & 0.322237 & 0.866489 \\ 0.769221 & 0.653489 & 0.229135 & 0.88469 & 0.82259 \\ 0.354298 & 0.384043 & 0.919686 & 0.648105 & 0.856111 \\ 0.315859 & 0.296394 & 0.434295 & 0.159272 & 0.108746 \\ 0.986888 & 0.728396 & 0.981009 & 0.277544 & 0.669064 \end{pmatrix}$$

Применяя схему Гаусса, приведем матрицу m к *треугольному* виду.

In[2]:= $\mathbf{Do[m[[i]] = m[[i]] - m[[j]]\ m[[i, j]]/m[[j, j]]//N, \{j, 1, 5\}, \{i, j + 1, 5\}]}$

In[3]:= $\mathbf{m//Chop}$

Out[3]=
$$\begin{pmatrix} 0.317823 & 0.53827 & 0.827947 & 0.322237 & 0.866489 \\ 0 & -0.649276 & -1.77473 & 0.104786 & -1.27456 \\ 0 & 0 & 0.587139 & 0.254026 & 0.3142 \\ 0 & 0 & 0 & -0.313484 & -0.425123 \\ 0 & 0 & 0 & 0 & 1.06755 \end{pmatrix}$$

Так как преобразования Гаусса не меняют определителя матрицы, а определитель треугольной матрицы равен произведению ее диагональных элементов, то определитель матрицы m равен

In[4]:= $\prod_{\mathbf{i=1}}^{\mathbf{5}} \mathbf{m[[i, i]]}$

Out[4]= 0.0405471

Для проверки введем команду

In[5]:= $\mathbf{Det[m1]}$

Out[5]= 0.0405471

§2 Обусловленность линейных алгебраических систем

Насколько решение системы чувствительно к неизбежным округлениям? Рассмотрим пример. Система

$$\begin{cases} x + 10y = 11, \\ 100x + 1001y = 1101 \end{cases} \tag{1}$$

имеет единственное решение $x = 1$, $y = 1$. Возмутим правую часть системы:

$$\begin{cases} x + 10y = 11.01, \\ 100x + 1001y = 1101. \end{cases} \tag{1}$$

Решение этой системы $x = 11.01$, $y = 0$. Можно сказать, что такая система не устойчива, так как малые изменение коэффициентов влекут сильные изменения решения. Какова мера неустойчивости линейной алгебраической системы?

Пусть

$$A.x = b \tag{1}$$

линейная алгебраическая система с невырожденной матрицей и не нулевым столбцом свободных членов b. Такая система имеет единственное решение x.

Допустим, что правая часть системы (1) получило возмущение $\triangle b$. Пусть $x+\triangle x$ решение возмущенной системы:

$$A.(x+\triangle x)=b+\triangle b. \tag{2}$$

Раскроем скобки и, учитывая, что x решение (1), получим

$$A.\triangle x=\triangle b.$$

Отсюда можно найти

$$\triangle x=A^{-1}.\triangle b,$$

а, применяя свойство нормы,

$$||\triangle x||\leqslant||A^{-1}||||\triangle b||. \tag{3}$$

Нормируя равенство (1), получим

$$||b||\leqslant||A||||x||. \tag{4}$$

Перемножим левые и правые части (3), (4), затем поделим части неравенства на $||b||||x||$ и получим

$$\frac{||\triangle x||}{||x||}\leqslant||A||\,||A^{-1}||\,\frac{||\triangle b||}{||b||}. \tag{5}$$

Здесь, отношение вида $\frac{||\triangle x||}{||x||}$ есть относительная погрешность для случая векторов.

Аналогичное неравенство получается, если возмутить элементы матрицы системы:

$$(A+\triangle A).(x+\triangle x)=b.$$

Раскроем скобки и, учитывая (1), получим, что

$$A.\triangle x+\triangle A.x+\triangle A.\triangle x=0,$$

а, пренебрегая малыми величинами $\triangle A.\triangle x$, получим

$$\triangle x=-A^{-1}.\triangle A.x.$$

Отсюда

$$||\triangle x||\leqslant||A^{-1}||\,||A||\,\frac{||\triangle A||}{||A||}\,||x||.$$

или, поделив на $||x||$,

$$\frac{||\triangle x||}{||x||}\leqslant||A^{-1}||\,||A||\,\frac{||\triangle A||}{||A||}. \tag{6}$$

Неравенства (5) и (6) позволяют оценить относительную погрешность решения в результате возмущения системы через погрешность коэффициентов системы.

Число $\nu(A)=||A||\cdot||A^{-1}||$ называется мерой обусловленности матрицы A.

Если норма $||E||=1$, то мера обусловленности ограничена снизу

$$\nu(A)=||A||\,||A^{-1}||\geqslant||A.A^{-1}||=||E||=1.$$

Если $\nu(A)$ велико, то матрица системы и сама система считаются плохо обусловленными. В этом случае из формул (5) и (6) следует, что относительная погрешность $\frac{||\triangle x||}{||x||}$ может быть велика, следовательно решения системы и возмущенной системы могут сильно отличаться, что и означает неустойчивость системы.

Вернемся к примеру. Матрица системы

$$A = \begin{pmatrix} 1 & 10 \\ 100 & 1001 \end{pmatrix}.$$

Обратная матрица

$$A^{-1} = \begin{pmatrix} 1001 & -10 \\ -100 & 1 \end{pmatrix}.$$

Рассмотрим следующую норму матрицы $A = (a_{i,j})$:

$$||A|| = \max_{i=1,\ldots,n} \sum_{j=1}^{n} |a_{ij}|.$$

Тогда $||A|| = 1101$, $||A^{-1}|| = 1011$, а мера обусловленности матрицы A

$$\nu(A) = 1101 \cdot 1011 > 10^6.$$

Так как

$$b = \begin{pmatrix} 11 \\ 1101 \end{pmatrix}, \quad \triangle b = \begin{pmatrix} 0.01 \\ 0 \end{pmatrix},$$

то $||b|| = 1101$, $|| \triangle b|| = 0.01$, а по формуле (5) получим

$$\frac{|| \triangle x||}{||x||} \leqslant \nu(A) \frac{|| \triangle b||}{||b||} = 1101 \cdot 1011 \cdot \frac{0.01}{1101} = 10.11. \tag{7}$$

Так как точное решение

$$x = \begin{pmatrix} 1 \\ 1 \end{pmatrix},$$

то $||x|| = 1$, а из (7) получим $|| \triangle x|| \leqslant 10.01$. Решение возмущенной системы

$$\begin{pmatrix} 11.01 \\ 0 \end{pmatrix}$$

вписывается в эту оценку $||x + \triangle x|| \leqslant ||x|| + || \triangle x|| \leqslant 1 + 10.01 = 11.01$.

Отметим, что близость невязки плохо обусловленной системы к нулю, не говорит о близости найденного приближенного решения к точному решению системы.

Предположим, что для системы примера получено два решения в результате разных округлений в процессе двух решений системы

$$x_1 = \begin{pmatrix} 11.01 \\ 0 \end{pmatrix} \text{ и } x_2 = \begin{pmatrix} 1 \\ 1.1 \end{pmatrix}.$$

Найдем невязку этих решений. Для первого решения $||r_1|| = ||b - A.x_1|| =$

$$|| \begin{pmatrix} 11 \\ 1101 \end{pmatrix} - \begin{pmatrix} 1 & 10 \\ 100 & 1001 \end{pmatrix} \cdot \begin{pmatrix} 11.01 \\ 0 \end{pmatrix} || = || \begin{pmatrix} -0.01 \\ 0 \end{pmatrix} || = 0.01.$$

Аналогично, находим невязку второго решения

$$||r_2|| = || \begin{pmatrix} 11 \\ 1101 \end{pmatrix} - \begin{pmatrix} 1 & 10 \\ 100 & 1001 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1.1 \end{pmatrix} || = || \begin{pmatrix} -1 \\ -100.1 \end{pmatrix} || = 100.1.$$

Тем не менее, второе решение ближе к точному решению!

§3 Метод прогонки

Методом прогонки решаются системы линейных алгебраических уравнений, матрицы которых имеют трехдиагональный вид. Дадим определение.

Матрица $A = (a_{i,j})$, $i, j = 1, 2, ..., n$, называется трехдиагональной, если в каждой строке все элементы отличные от $a_{i,i-1}$, $a_{i,i}$, $a_{i,i+1}$ - нулевые. Например, при $n = 5$, трехдиагональная матрица имеет следующий вид:

$$\begin{pmatrix} * & * & 0 & 0 & 0 \\ * & * & * & 0 & 0 \\ 0 & * & * & * & 0 \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{pmatrix}.$$

Системы с трехдиагональной матрицей возникают при построении сплайнов или решении дифференциальных уравнений разностным методом и называются системами разностных уравнений. Метод прогонки боле экономичен, чем другие, например, метод Гаусса. Если n - размерность системы, то метод прогонки позволяет получить решение примерно за $8n$ арифметических операций.

Рассмотрим систему уравнений

$$\begin{cases} c_1x_1+ & d_1x_2 & . & . & . & . & . & . & = & r_1, \\ b_2x_1+ & c_2x_2+ & d_2x_3 & . & . & . & . & . & = & r_2, \\ . & b_3x_2+ & c_3x_3+ & d_3x_4 & . & . & . & . & = & r_3, \\ . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & b_{n-1}x_{n-2}+ & c_{n-1}x_{n-1}+ & d_{n-1}x_n & = & r_{n-1}, \\ . & . & . & . & . & . & b_nx_{n-1}+ & c_nx_n & = & r_n \end{cases} \tag{1}$$

с трехдиагональной расширенной матрицей:

$$\left(\begin{array}{cccccccc|c} c_1 & d_1 & 0 & 0 & . & 0 & 0 & 0 & r_1 \\ b_2 & c_2 & d_2 & 0 & . & 0 & 0 & 0 & r_2 \\ 0 & b_3 & c_3 & d_3 & . & 0 & 0 & 0 & r_3 \\ & & . & . & . & . & . & . & \\ 0 & 0 & 0 & 0 & . & b_{n-1} & c_{n-1} & d_{n-1} & r_{n-1} \\ 0 & 0 & 0 & 0 & . & 0 & b_n & c_n & r_n \end{array}\right).$$

Выразим x_i через x_{i+1}. Будем искать такое выражение в виде

$$x_i = \delta_i\, x_{i+1} + \lambda_i, \quad i = 1, 2, ..., n-1. \tag{2}$$

Найдем для каждого i коэффициенты прогонки δ_i и λ_i. Из первого уравнения (1) находим

$$x_1 = -\frac{d_1}{c_1}x_2 + \frac{r_1}{c_1}. \tag{3}$$

Следовательно

$$\delta_1 = -\frac{d_1}{c_1}, \qquad \lambda_1 = \frac{r_1}{c_1}. \tag{$*$}$$

Выражение x_1 из (3) подставим во второе уравнение (1), получим

$$b_2(\delta_1 x_2 + \lambda_1) + c_2 x_2 + d_2 x_3 = r_2.$$

Отсюда находим

$$x_2 = -\frac{d_2}{b_2\delta_1 + c_2} x_3 + \frac{r_2 - b_2\lambda_1}{b_2\delta_1 + c_2}.$$

Сравнивая с (2) при $i = 2$, получаем коэффициенты

$$\delta_2 = -\frac{d_2}{b_2\delta_1 + c_2}, \qquad \lambda_2 = \frac{r_2 - b_2\lambda_1}{b_2\delta_1 + c_2}.$$

Найденное выражение для x_2 подставим в третье уравнение (1), найдем коэффициенты δ_3, λ_3 и так далее. Подставив $x_{i-1} = \delta_{i-1}x_i + \lambda_{i-1}$ в i-ое уравнение (1), получим

$$b_i(\delta_{i-1}x_i + \lambda_{i-1}) + c_i x_i + d_i x_{i+1} = r_i.$$

Отсюда находим

$$x_i = \delta_i\, x_{i+1} + \lambda_i, \tag{4}$$

где

$$\delta_i = -\frac{d_i}{b_i\delta_{i-1} + c_i}, \qquad \lambda_i = \frac{r_i - b_i\lambda_{i-1}}{b_i\delta_{i-1} + c_i}; \tag{$**$}$$

здесь $i = 2, ..., n-1$. При $i = n-1$ равенство (4) примет вид

$$x_{n-1} = \delta_{n-1}x_n + \lambda_{n-1},$$

где коэффициенты вычисляются по формулам $(**)$. Подставим такое выражение x_{n-1} в последнее уравнение (1), получим

$$b_n(\delta_{n-1}x_n + \lambda_{n-1}) + c_n x_n = r_n.$$

Отсюда найдем

$$x_n = \frac{r_n - b_n\lambda_{n-1}}{b_n\delta_{n-1} + c_n}. \tag{$***$}$$

Решение системы (1) методом прогонки делится на две части - прямая прогонка и обратная.

Прямая прогонка позволяет вычислять коэффициенты прогонки по формулам $(*)$ и $(**)$ при $i = 2, 3, ..., n-1$.

Обратная прогонка. Зная прогоночные коэффициенты, находим решения системы. По формуле $(***)$ находим x_n, а из формулы (2) находим остальные неизвестные x_{n-1}, $x_{n-2}, \ldots\ x_1$.

Прогонка называется корректной, если знаменатели в $(*)$, $(**)$ и $(***)$ отличны от нуля и устойчивой, если $|\delta_i| < 1$ для всех i=1,2,...,n.

Теорема. Если трехдиагональная матрица системы уравнений (1) есть матрица с преобладанием диагональных элементов, то есть

$$|c_i| > |b_i| + |d_i|, \quad i = 1, 2, ..., n, \tag{5}$$

то прогонка корректна и устойчива. Здесь считается, что $b_1 = d_n = 0$.

Доказательство проведем методом математической индукции. При $i = 1$ из (5) получим $|c_1| > |d_1| \geqslant 0$. Значит знаменатель в $(*)$ отличен от нуля и $|\delta_1| = |-\frac{d_1}{c_1}| < 1$.

Пусть знаменатель $i-1$ коэффициентов в формулах $(**)$ отличен от нуля и справедливо неравенство $|\delta_{i-1}| < 1$. Покажем, что знаменатели i-х коэффициентов в формулах $(**)$ отличен от нуля и $|\delta_i| < 1$. Применяя (5), находим оценку знаменателя в формулах $(**)$ для i-х коэффициентов: $|b_i\delta_{i-1} + c_i| \geqslant |c_i| - |b_i||\delta_{i-1}| > |b_i| + |d_i| - |b_i||\delta_{i-1}| = |d_i| + |b_i|(1 - |\delta_{i-1}|) \geqslant |d_i| \geqslant 0$. Что говорит о корректности прогонки. Кроме того, $|\delta_i| = 0 < 1$, если $d_i = 0$, а при $d_i \neq 0$

$$|\delta_i| = \Big| -\frac{d_i}{b_i\delta_{i-1} + c_i}\Big| = \frac{|d_i|}{|b_i\delta_{i-1} + c_i|} < \frac{|d_i|}{|d_i|} = 1.$$

Следовательно прогонка устойчива. Теорема доказана.

Замечания. 1). Система (1) имеет единственное решение при выполнении условия (5), так как матрица с преобладанием диагональных элементов имеет ненулевой определитель.

2). Число арифметических операций для нахождения прогоночных коэффициентов равно $6(n-2)+2+5$, для определения неизвестных - $2(n-1)$. Таким образом, всего арифметических операций в методе прогонки $8n - 7 \approx 8n$.

Подготовка задания для метода прогонки. Покажем как построить трехдиагональную матрицу системы и проверить корректность прогонки.

Можно ввести необязательную команду округления числа до n знаков после запятой. Эта команда позволит более компактно выводить матрицы на экран.

In[1]:=
```
rn[t_, n_] := Round[10^n t]/10^n//N
```

Генерируем расширенную матрицу системы, например, порядка 5×6. Пусть элементы матрицы положительны. Сделаем диагональные элементы матрицы больше, чем сумма всех остальных элементов в строке. Тогда матрица системы будет удовлетворять условию теоремы, то есть, будет матрицей с преобладанием диагональных элементов.

In[2]:=
```
m = Table[
      If[i == j, Random[Integer, {2, 5}] + Random[],
      Random[]],
    {i, 1, 5}, {j, 1, 6}]//N
```

Округлим все элементы матрицы до третьего знака после запятой.

In[3]:=
```
m = rn[m, 3]
```

Out[3]=

$$\begin{pmatrix} 2.192 & 0.649 & 0.096 & 0.446 & 0.55 & 0.132 \\ 0.385 & 4.002 & 0.862 & 0.584 & 0.858 & 0.182 \\ 0.149 & 0.209 & 5.886 & 0.032 & 0.634 & 0.641 \\ 0.607 & 0.689 & 0.115 & 4.89 & 0.002 & 0.383 \\ 0.923 & 0.241 & 0.906 & 0.937 & 3.373 & 0.109 \end{pmatrix}$$

Сделаем из матрицы **m** трехдиагональную матрицу, обнуляя ее соответствующие элементы:

In[4]:=
```
Do[If[j > i + 1&&j! = 6||i > j + 1&&j! = 6, m[[i, j]] = 0],
  {i, 1, 5}, {j, 1, 6}]
```

Проверим результат выполнение операции:

In[5]:=
```
m//MatrixForm
```

Out[5]=

$$\begin{pmatrix} 2.192 & 0.649 & 0 & 0 & 0 & 0.132 \\ 0.385 & 4.002 & 0.862 & 0 & 0 & 0.182 \\ 0 & 0.209 & 5.886 & 0.032 & 0 & 0.641 \\ 0 & 0 & 0.115 & 4.89 & 0.002 & 0.383 \\ 0 & 0 & 0 & 0.937 & 3.373 & 0.109 \end{pmatrix}$$

В результате получили расширенную матрицу системы, удовлетворяющей условию (5).

Проверка корректности прогонки. Преобладание диагональных элементов матрицы системы означает корректность прогонки. Следующая команда проверяет преобладание диагональных элементов матрицы m. Сначала проверяется условие

$$Abs[m[[i,i]]] > Abs[m[[i,i-1]]] + Abs[m[[i,i+1]]]$$

для каждого $i = 2, 3, 4$. Каждое неравенство возвращает $True$, если условие верно или $False$ в противном случае. Все такие значения собираются в список командой $Table$. Команда $MemberQ$ возвращает $False$, если список состоит из одних значений $True$, а отрицание Not этой команды, в этом случае, вернет $True$. Если при этом выполняются неравенства для первой и последней строки $m[[1,1]] > m[[1,2]]$ и $m[[5,5]] > m[[5,4]]$, то первый аргумент в команде If будет $True$ и появится сообщение о корректности прогонки. Если хоть одно из перечисленных неравенств не выполняется, то прогонка не корректна.

In[6]:=
```
korprog[m_] :=
 If[
   Not[MemberQ[
     Table[Abs[m[[i, i]]] > Abs[m[[i, i - 1]]] + Abs[m[[i, i + 1]]],
       {i, 2, 4}], False]] &&
   m[[1, 1]] > m[[1, 2]] && m[[5, 5]] > m[[5, 4]],
   Print["Прогонка корректна."],
   Print["Условие не выполняется."]
 ]
```

Проверим, корректна ли прогонка для системы с расширенной матрицей m

In[7]:=
```
korprog[m]
```

Out[7]= Прогонка корректна.

Решим теперь систему с расширенной матрицей m методом прогонки.

Прямая прогонка.

Находим коэффициенты

In[10]:=
$$\delta_1 = -\frac{m[[1,2]]}{m[[1,1]]};$$

In[11]:=
$$\lambda_1 = \frac{m[[1,6]]}{m[[1,1]]};$$

и

In[12]:=
$$\text{Do}\Big[\delta_i = -\frac{m[[i,i+1]]}{m[[i,i-1]]\delta_{i-1} + m[[i,i]]};\ \lambda_i = \frac{m[[i,6]] - m[[i,i-1]]\,\lambda_{i-1}}{m[[i,i-1]]\delta_{i-1} + m[[i,i]]},\ \{i,2,5\}\Big]$$

Обратная прогонка.

Находим неизвестную

In[13]:= $\mathbf{x_5 = \lambda_5}$

Out[13]= 0.0441191

и остальные неизвестные

In[14]:= $\mathbf{Table[x_i = \delta_i\, x_{i+1} + \lambda_i, \{i, 4, 1, -1\}]}$

Out[14]= {0.0770222, 0.152198, 0.143964, -0.0212256}

Вектор решения

In[15]:= $\mathbf{x = Table[x_i, \{i, 1, 5\}]}$

Out[15]= {-0.0212256, 0.143964, 0.152198, 0.0770222, 0.0441191}

Столбец свободных членов системы уравнений

In[16]:= $\mathbf{bb = m[[All, -1]]}$

Out[16]= {0.0378916, 0.517047, 0.819762, 0.567489, 0.305136}

Матрица системы

$\mathbf{(a = m[[1;;, 1;;5]])//MatrixForm}$

Out[16]=

$$\begin{pmatrix} 2.192 & 0.649 & 0 & 0 & 0 \\ 0.385 & 4.002 & 0.862 & 0 & 0 \\ 0 & 0.209 & 5.886 & 0.032 & 0 \\ 0 & 0 & 0.115 & 4.89 & 0.002 \\ 0 & 0 & 0 & 0.937 & 3.373 \end{pmatrix}$$

Проверяем невязку

In[17]:= $\mathbf{bb - a.x//Chop}$

Out[16]= {0, 0, 0, 0, 0}

Составим команду, реализующую метод прогонки для системы произвольной размерности. Сначала изменим команду проверки корректности прогонки, напишем ее для произвольной размерности n:

In[6]:=
```
korprog[m_] :=
 Block[{n = Length[m]},
  If[
   Not[MemberQ[
    Table[
     Abs[m[[i, i]]] > Abs[m[[i, i - 1]]] + Abs[m[[i, i + 1]]],
     {i, 2, n - 1}],
     False]]&&
    m[[1, 1]] > m[[1, 2]]&&m[[n, n]] > m[[n, n - 1]],
    Print["Прогонка корректна."],
    Print["Условие не выполняется."]
   ]
 ]
```

и введем основную функцию, реализующую прогонку:

In[18]:=
```
prog[m1_] := Block[{m, n = Length[m1]},
  m = m1;
  bb = m[[All, -1]];
  a = m[[1;;, 1;;n]];
  δ1 = -m[[1, 2]]/m[[1, 1]];
```

$\lambda_1 = \frac{m[[1, n+1]]}{m[[1,1]]}$;
Do[
$\delta_i = -\frac{m[[i, i+1]]}{m[[i, i-1]]\delta_{i-1} + m[[i,i]]}$;
$\lambda_i = \frac{m[[i, n+1]] - m[[i, i-1]]\lambda_{i-1}}{m[[i, i-1]]\delta_{i-1} + m[[i,i]]}$,
{i, 2, n}];
$x_n = \lambda_n$;
Do[
$x_i = \delta_i x_{i+1} + \lambda_i$,
{i, n − 1, 1, −1}];
x = Table[x_i, {i, 1, n}];
Print["Ответ : x = ", x, "с невязкой r = ", bb − a.x//Chop]]

Применение команды при введенной расширенной матрицы системы:

In[19]:= **prog[m]**

Out[19]= Ответ: x={0.0552071, 0.0169277, 0.107889, 0.0757812, 0.0112639}
с невязкой r={0, 0, 0, 0, 0}

Замечание. Если в команде prog вместо Print["Ответ: x= ", x, "с невязкой r=", bb - a . x // Chop] поставить x, то команда будет возвращать только ответ. Команду в таком виде можно применять в других программах.

§4 LU-разложение матрицы

Представим квадратную матрицу $A = (a_{ij})$ в виде произведения двух матриц, L - нижней треугольной с единичными диагональными элементами и U - верхней треугольной матриц: $A = L \cdot U$.

Например, если матрица A имеет порядок $n = 5$, то найдем разложение вида:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 & 0 \\ l_{41} & l_{42} & l_{43} & 1 & 0 \\ l_{51} & l_{52} & l_{53} & l_{54} & 1 \end{pmatrix} \cdot \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} & u_{15} \\ 0 & u_{22} & u_{23} & u_{24} & u_{25} \\ 0 & 0 & u_{33} & u_{34} & u_{35} \\ 0 & 0 & 0 & u_{44} & u_{45} \\ 0 & 0 & 0 & 0 & u_{55} \end{pmatrix}$$

Здесь, первая матрица справа есть нижняя треугольная матрица с единицами на главной диагонали, а вторая матрица - верхняя треугольная. Перемножим матрицы, стоящие справа в этом равенстве и, приравнивая соответствующие элементы матрицы A и произведения матриц, получим систему $n^2 = 25$ уравнений для определения n^2 неизвестных u_{mi}, и l_{jm} :

$$\begin{array}{lll} u_{11} = a_{11}, & u_{12} = a_{12}, & u_{13} = a_{13}, \\ l_{21}u_{11} = a_{21}, & l_{21}u_{12} + u_{22} = a_{22}, & l_{21}u_{13} + u_{23} = a_{23}, \\ l_{31}u_{11} = a_{31}, & l_{31}u_{12} + l_{32}u_{22} = a_{32}, & l_{31}u_{13} + l_{32}u_{23} + u_{33} = a_{33}, \\ l_{41}u_{11} = a_{41}, & l_{41}u_{12} + l_{42}u_{22} = a_{42}, & l_{41}u_{13} + l_{42}u_{23} + l_{43}u_{33} = a_{43}, \\ l_{51}u_{11} = a_{51}, & l_{51}u_{12} + l_{52}u_{22} = a_{52}, & l_{51}u_{13} + l_{52}u_{23} + l_{53}u_{33} = a_{53}, \end{array}$$

$$\begin{array}{ll} u_{14} = a_{14}, & u_{15} = a_{15} \\ l_{21}u_{14} + u_{24} = a_{24}, & l_{21}u_{15} + u_{25} = a_{25}, \\ l_{31}u_{14} + l_{32}u_{24} + u_{34} = a_{34}, & l_{31}u_{15} + l_{32}u_{25} + u_{35} = a_{35}, \\ l_{41}u_{14} + l_{42}u_{24} + l_{43}u_{34} + u_{4,4} = a_{44}, & l_{41}u_{15} + l_{42}u_{25} + l_{43}u_{35} + u_{45} = a_{45}, \\ l_{51}u_{14} + l_{52}u_{24} + l_{53}u_{34} + l_{54}u_{4,4} = a_{54}, & l_{51}u_{15} + l_{52}u_{25} + l_{53}u_{35} + l_{54}u_{45} + u_{55} = a_{55} \end{array}$$

Из первых строк находим элементы u_{1i} :

$$u_{1i} = a_{1i}, \quad i = 1, ..., n \tag{1}$$

Мысленно вычеркнем первые строки из всех столбцов. Из оставшихся элементов первого столбца найдем элементы l_{j1} :

$$l_{j1} = \frac{a_{j1}}{u_{11}}, \quad i = 2, ..., n \tag{1'}$$

Мысленно вычеркнем первый столбец из этой системы уравнений. Из оставшихся элементов вторых строк найдем элементы u_{2i} :

$$u_{2i} = a_{2i} - l_{21}u_{1i}, \quad i = 2, ..., n \tag{2}$$

Вычеркнем вторые строки из всех столбцов. Из оставшихся элементов второго столбца найдем элементы l_{j2} :

$$l_{j2} = \frac{1}{u_{22}}(a_{j2} - l_{j1}u_{12}), \quad i = 3, ..., n \tag{2'}$$

Вычеркнем второй столбец из этой системы уравнений. Из оставшихся элементов третьих строк найдем элементы u_{3i} :

$$u_{3i} = a_{3i} - l_{31}u_{1i} - l_{32}u_{2i} = a_{3i} - \sum_{k=1}^{2} l_{3k}u_{ki}, \quad i = 3, ..., n \tag{3}$$

Вычеркнем третьи строки из всех столбцов. Из оставшихся элементов третьего столбца найдем элементы l_{j3} :

$$l_{j3} = \frac{1}{u_{33}}(a_{j3} - l_{j1}u_{13} - l_{j2}u_{23}) = \frac{1}{u_{33}}(a_{j3} - \sum_{k=1}^{2} l_{jk}u_{k3}), \quad i = 3, ..., n \tag{3'}$$

И так далее, будем поочередно находить элементы u_{mi}, и l_{jm}. Обобщая формулы (2), (3) и (2'), (3'), можно записать, что

$$u_{mi} = a_{mi} - \sum_{k=1}^{m-1} l_{mk}u_{ki}, \quad m = 2, ..., n, \;\; i = m, ..., n, \tag{4}$$

$$l_{jm} = \frac{1}{u_{mm}}(a_{jm} - \sum_{k=1}^{m-1} l_{jk}u_{km}), \quad m = 2, ..., n, \;\; j = m+1, ..., n. \tag{4'}$$

При этом будем считать, что сумма в этих равенствах равна нулю, если $m = 1$.

Таким образом, элементы матриц-сомножителей определяются формулами (1), (1') и (4), (4').

Теорема. Пусть A - квадратная матрица порядка n, A_k - матрица главного минора матрицы A, составленная из элементов первых k-строк и k-столбцов. Если $\det A_k \neq 0$ для всех $k = 1, 2, ..., n$, то существует единственная нижняя треугольная матрица $L = (l_{ij})$ с $l_{11} = l_{22} = ... = l_{nn} = 1$ и единственная верхняя треугольная матрица $U = (u_{ij})$ такие, что

$$A = L.U. \tag{5}$$

Доказательство проведем по методу математической индукции. Пусть $n = 1$. Тогда $A = (a_{11})$. Если $L = (1)$, $U = (u_{11})$, то из равенства $A = L.U$ следует, что $a_{11} = u_{11}$. Пусть теорема верна для всех таких матриц A порядка $k-1$. Докажем утверждение теоремы для матрицы A порядка k. Запишем матрицу A порядка k в блочном виде:

$$A_k = \begin{pmatrix} A_{k-1} & | & x \\ -- & -- & -- \\ y & | & a_{kk} \end{pmatrix}.$$

Введем матрицы - сомножители:

$$L_k = \begin{pmatrix} L_{k-1} & | & 0 \\ -- & -- & -- \\ z & | & 1 \end{pmatrix}, \quad U_k = \begin{pmatrix} U_{k-1} & | & u \\ -- & -- & -- \\ 0 & | & u_{kk} \end{pmatrix}.$$

Для определения блоков L_{k-1}, U_{k-1}, векторов z, u и элемента u_{kk} запишем равенство (5). Так как

$$L_k.U_k = \begin{pmatrix} L_{k-1}.U_{k-1} & | & L_{k-1}.u \\ -- & -- & -- \\ z.U_{k-1} & | & zu + u_{kk} \end{pmatrix}.$$

то из равенства (5), записанного для этих матриц $A_k = L_k.U_k$, получаем:

$$A_{k-1} = L_{k-1}.U_{k-1}, \;\; x = L_{k-1}.u, \;\; y = z.U_{k-1}, \;\; a_{kk} = zv + u_{kk}.$$

По предположению индукции существуют и единственны матрицы L_{k-1} и U_{k-1} такие, что $A_{k-1} = L_{k-1}.U_{k-1}$. Этим определяется первый блок матрицы A_k. Так как $\det A_{k-1} \neq 0$, то и $\det L_{k-1} \neq 0$, $\det U_{k-1} \neq 0$. Следовательно решения систем $x = L_{k-1}.u$ и $y = z.U_{k-1}$ существуют и единственны - u, z. Отсюда находим и $u_{kk} = a_{kk} - zv$. Теорема доказана.

В соответствии с доказанной теоремой проверку разложения матрицы A в произведение $L.U$ можно провести следующей командой

```
In[1]:= proverkaLU[b_] := Block[{q, n = Length[a]},
          q = Table[a[[1;;k, 1;;k]], {k, 1, n}];
          If[MemberQ[Det/@q, 0],
          Print["Разложение не существует"],
          Print["LU разложение существует"]]]
```

Команда *Table* формирует список q из матриц главных миноров матрицы a. Команда $MemberQ[Det/@q, 0]$ проверяет вхождение 0 в список миноров q и возвращает True, если вхождения есть. Таким образом, если команда $MemberQ[Det/@q, 0]$ вернет True, то условие теоремы не выполняется для матрицы b. Команда $proverkaLU[b]$ вернет предложение: "Разложение не существует".

n.1 Решение систем линейных уравнений методом LU-разложения

Предположим, что матрица A системы линейных уравнений

$$A.x = b \tag{6}$$

допускает LU-разложение: $A = L.U$. Система (6) примет вид $L.U.x = b$. Если обозначить $y = U.x$, то $L.y = b$ и решение x системы (6) можно найти в два этапа: сначала решить систему $L.y = b$, а потом систему $U.x = y$. Так как матрицы этих систем имеют треугольный вид, то их решение находится аналогично обратному ходу метода Гаусса.

Рассмотрим конкретный пример. Найдем решение системы линейных уравнений с матрицей A :

In[1]:= $\mathbf{n = 5;}$

In[2]:= $\mathbf{A = Table[a_{i,j} = i^j, \{i, 1, n\}, \{j, 1, n\}];}$

Out[2]=

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 2 & 4 & 8 & 16 & 32 \\ 3 & 9 & 27 & 81 & 243 \\ 4 & 16 & 64 & 256 & 1024 \\ 5 & 25 & 125 & 625 & 3125 \end{pmatrix}$$

и столбцом свободных членов

In[3]:= $\mathbf{b = \{1, 3, 2, 5, 4\};}$

Отметим, что при задании матрицы A перечислением ее элементов, надо определить $a_{i,j}$, например, так, как это сделано выше, или вместо $a_{i,j}$ писать $A[[i, j]]$.

Найдем LU разложение матрицы матрицы A.

Проверим существование такого разложения, воспользуемся введенной ранее командой

In[4]:= $\mathbf{proverkaLU[A]}$

Out[4]=

LU разложение существует.

Введем матрицы L, пока с нулевыми диагональными элементами, и U :

In[5]:= $\mathbf{L = Table[If[i > j, l_{i,j}, 0], \{i, 1, n\}, \{j, 1, n\}]}$

$\mathbf{U = Table[If[i \leqslant j, u_{i,j}, 0], \{i, 1, n\}, \{j, 1, n\}]}$

Out[5]= $L = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ l_{21} & 0 & 0 & 0 & 0 \\ l_{31} & l_{32} & 0 & 0 & 0 \\ l_{41} & l_{42} & l_{43} & 0 & 0 \\ l_{51} & l_{52} & l_{53} & l_{54} & 0 \end{pmatrix}$ $U = \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} & u_{15} \\ 0 & u_{22} & u_{23} & u_{24} & u_{25} \\ 0 & 0 & u_{33} & u_{34} & u_{35} \\ 0 & 0 & 0 & u_{44} & u_{45} \\ 0 & 0 & 0 & 0 & u_{55} \end{pmatrix}$

Диагональные элементы матрицы L сделаем единичными:

In[6]:= $\mathbf{Do[L[[i, i]] = 1, \{i, 1, n\}]; L}$

Out[6]=

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 & 0 \\ l_{41} & l_{42} & l_{43} & 1 & 0 \\ l_{51} & l_{52} & l_{53} & l_{54} & 1 \end{pmatrix}$$

Применяя формулы (1) и (1'), найдем первую строку матрицы U и первый столбец матрицы L :

In[7]:=
$$\mathbf{Do}[\mathbf{u_{1,j}} = \mathbf{a_{1,j}}, \{\mathbf{j}, \mathbf{1}, \mathbf{n}\}]$$
$$\mathbf{Do}[\mathbf{l_{i,1}} = \frac{\mathbf{a_{i,1}}}{\mathbf{u_{1,1}}}, \{\mathbf{i}, \mathbf{2}, \mathbf{n}\}]$$

Найдем вторую строку матрицы U и второй столбец матрицы L, применяя формулы (4) и (4'), при

In[8]:=
$$\mathbf{m} = \mathbf{2};$$
$$\mathbf{Do}[\mathbf{u_{m,i}} = \mathbf{a_{m,i}} - \sum_{\mathbf{k=1}}^{\mathbf{m-1}} \mathbf{l_{m,k}}\mathbf{u_{k,i}}, \{\mathbf{i}, \mathbf{m}, \mathbf{n}\}];$$
$$\mathbf{Do}[\mathbf{l_{j,m}} = \frac{\mathbf{1}}{\mathbf{u_{m,m}}}(\mathbf{a_{j,m}} - \sum_{\mathbf{k=1}}^{\mathbf{m-1}} \mathbf{l_{j,k}}\mathbf{u_{k,m}}), \{\mathbf{j}, \mathbf{m+1}, \mathbf{n}\}];$$

Полагая

In[9]:=
$$\mathbf{m} = \mathbf{3};$$

выполним предыдущие две команды еще раз, затем выполним эти же команды при $m = 4, 5$ и найдем все элементы матриц L и U.

Можно применить команду цикла и объединить эти команды, то есть, вместо команд вида 8 выполнить следующую команду цикла:

$$\mathbf{Do}[$$
$$\mathbf{Do}[\mathbf{u_{m,i}} = \mathbf{a_{m,i}} - \sum_{\mathbf{k=1}}^{\mathbf{m-1}} \mathbf{l_{m,k}}\mathbf{u_{k,i}}, \{\mathbf{i}, \mathbf{m}, \mathbf{n}\}];$$
$$\mathbf{Do}[\mathbf{l_{j,m}} = \frac{\mathbf{1}}{\mathbf{u_{m,m}}}(\mathbf{a_{j,m}} - \sum_{\mathbf{k=1}}^{\mathbf{m-1}} \mathbf{l_{j,k}}\mathbf{u_{k,m}}), \{\mathbf{j}, \mathbf{m+1}, \mathbf{n}\}],$$
$$\{\mathbf{m}, \mathbf{2}, \mathbf{n}\}]$$

В результате получим матрицы:

In[11]:=
$$\mathbf{L}$$
$$\mathbf{U}$$

Out[11]=
$$L = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 \\ 3 & 3 & 1 & 0 & 0 \\ 4 & 6 & 4 & 1 & 0 \\ 5 & 10 & 10 & 5 & 1 \end{pmatrix} \qquad U = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 2 & 6 & 14 & 30 \\ 0 & 0 & 6 & 36 & 150 \\ 0 & 0 & 0 & 24 & 240 \\ 0 & 0 & 0 & 0 & 120 \end{pmatrix}$$

Решаем теперь систему $L.y = b$:

$$\begin{cases} y_1 == 1, \\ 2y_1 + y_2 == 3, \\ 3y_1 + 3y_2 + y_3 == 2, \\ 4y_1 + 6y_2 + 4y_3 + y_4 == 5, \\ 5y_1 + 10y_2 + 10y_3 + 5y_4 + y_5 == 4. \end{cases}$$

Выполним обратных ход. Он такой же как и в методе Гаусса.

In[12]:=
$$\mathbf{Table}[\mathbf{y_i} = \mathbf{b}[[\mathbf{i}]] - \sum_{\mathbf{j=1}}^{\mathbf{i}} \mathbf{If}[\mathbf{i} == \mathbf{j}, \mathbf{0}, \mathbf{L}[[\mathbf{i}, \mathbf{j}]]]\mathbf{y_j}, \{\mathbf{i}, \mathbf{1}, \mathbf{n}\}]$$

Out[12]= {1, 1, -4, 11, -26}

Решаем теперь систему $U.x = y$:

$$\begin{cases} x_1 + x_2 + x_3 + x_4 + x_5 == 1, \\ 2x_2 + 6x_3 + 14x_4 + 30x_5 == 1, \\ 6x_3 + 36x_4 + 150x_5 == -4, \\ 24x_4 + 240x_5 == 11, \\ 120x_5 == -26. \end{cases}$$

По аналогии с обратным ходом метода Гаусса, получим

```
In[13]:= xo = Table[x_i = (y_i - Sum_{j=i}^{n} If[i == j, 0, U[[i, j]]] x_j)/U[[i, i]],
            {i, n, 1, -1}]
Out[13]= {-13/60, 21/8, -11, 147/8, -527/60}
```

Окончательный ответ:

```
In[14]:= xo = Reverse[xo]
Out[14]= {-527/60, 147/8, -11, 21/8, -13/60}
```

Собрав все команды, получим программы, реализующие LU разложение матрицы a с выводом ответов на экран и решение матричного уравнения $a.x = b$ при помощи LU разложения.

```
In[1]:= LUdecomposition[a_] :=
          Block[{n = Length[a]},
            U = Table[If[i ≤ j, u_{i,j}, 0], {i, 1, n}, {j, 1, n}];
            L = Table[If[i ≥ j, If[i == j, 1, l_{i,j}], 0], {i, 1, n}, {j, 1, n}];
            Do[u_{1,j} = a[[1, j]], {j, 1, n}];
            Do[l_{i,1} = a[[i, 1]]/u_{1,1}, {i, 2, n}];
            Do[
              Do[u_{m,i} = a[[m, i]] - Sum_{k=1}^{m-1} l_{m,k} u_{k,i}, {i, m, n}];
              Do[l_{j,m} = 1/u_{m,m} (a[[j, m]] - Sum_{k=1}^{m-1} l_{j,k} u_{k,m}), {j, m + 1, n}],
            {m, 2, n}];
            Print["L = ", L, ", ", "U = ", U]]
In[2]:= LUsolveslau[a_, b_] :=
          Block[{n = Length[a]},
            Table[y_i = b[[i]] - Sum_{j=1}^{i} If[i == j, 0, L[[i, j]]] y_j,
              {i, 1, n}];
            xo = Table[x_i = (y_i - Sum_{j=i}^{n} If[i == j, 0, U[[i, j]]] x_j)/U[[i, i]],
              {i, n, 1, -1}];
            xo = Reverse[xo];
            Print["x = ", xo//N]]
```

Напомним, что команда LUsolveslau вернет только ответ, если вместо команды Print["x=", xo//N] записать только xo или xo//N.

Команда *LUdecomposition*[a] без вывода матриц на экран понадобится в следующем пункте.

n.2 Вычисление определителей и обращение матриц с помощью LU-разложения

1. Если матрица $A = (a_{ij})$ допускает LU разложение $A = L \cdot U$, то, $\det A = \det L \cdot \det U$. Так, как $\det L = 1$, $\det U = u_{11}u_{22}...u_{nn}$, то:

$$\det A = u_{11}u_{22}...u_{nn}.$$

2. Пусть $A^{-1} = (x_{ij})$ - обратная матрица для матрицы $A = (a_{ij})$. Получим явные формулы для вычисления элементов x_{ij} обратной матрицы, при условии, что известно LU-разложение матрицы A.

Из равенства $A = L \cdot U$ получим равенство $A^{-1} = U^{-1} \cdot L^{-1}$. Умножая последнее равенство на U слева и на L справа, получим равенства:

$$U \cdot A^{-1} = L^{-1}, \qquad A^{-1} \cdot L = U^{-1}.$$

Последние соотношения, например при $n = 4$, можно записать так:

$$\begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{pmatrix} \cdot \begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ * & 1 & 0 & 0 \\ * & * & 1 & 0 \\ * & * & * & 1 \end{pmatrix},$$

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{pmatrix} = \begin{pmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{pmatrix}.$$

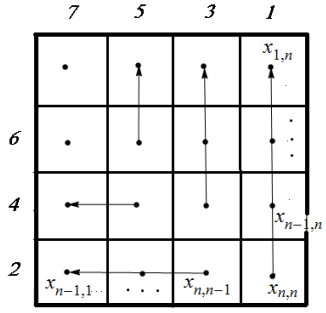
где " $*$ " обозначены соответствующие элементы матриц.

Перемножив матрицы и приравняв те элементы матриц, которые не содержат звездочек, получим в общем случае $n^2 = 16$ уравнений относительно 16 неизвестных элементов $x_{i,j}$ обратной матрицы A^{-1}.

$$\begin{array}{ll} u_{11}x_{11} + u_{12}x_{21} + u_{13}x_{31} + u_{14}x_{41} = 1, & u_{11}x_{12} + u_{12}x_{22} + u_{13}x_{32} + u_{14}x_{42} = 0, \\ x_{21} + l_{21}x_{22} + l_{31}x_{23} + l_{41}x_{24} = 0, & u_{22}x_{22} + u_{23}x_{32} + u_{24}x_{42} = 1, \\ x_{31} + l_{21}x_{32} + l_{31}x_{33} + l_{41}x_{34} = 0, & x_{32} + l_{32}x_{33} + l_{42}x_{34} = 0, \\ x_{41} + l_{21}x_{42} + l_{31}x_{43} + l_{41}x_{44} = 0, & x_{42} + l_{32}x_{43} + l_{42}x_{44} = 0, \end{array}$$

$$\begin{array}{ll} u_{11}x_{13} + u_{12}x_{23} + u_{13}x_{33} + u_{14}x_{43} = 0, & u_{11}x_{14} + u_{12}x_{24} + u_{13}x_{34} + u_{14}x_{44} = 0, \\ u_{22}x_{23} + u_{23}x_{33} + u_{24}x_{43} = 0, & u_{22}x_{24} + u_{23}x_{34} + u_{24}x_{44} = 0, \\ u_{33}x_{33} + u_{34}x_{43} = 1, & u_{33}x_{34} + u_{34}x_{44} = 0, \\ x_{43} + l_{43}x_{44} = 0, & u_{44}x_{44} = 1. \end{array}$$

Решение этой системы проводится по следующей графической подсказке



Находим $x_{n,n}$ $(n = 4)$ из последнего уравнения последнего столбца:

$$x_{n,n} = \frac{1}{u_{n,n}}. \tag{1}$$

Из уравнений последнего столбца, определяем $x_{n-1,n}$, $x_{n-2,n}, ...,$ $x_{1,n}$:

$$x_{i,n} = -\frac{1}{u_{i,i}} \sum_{k=i+1}^{n} u_{i,k} x_{k,n}, \quad i = n-1, n-2, ..., 1. \tag{2}$$

Из уравнений последних строк определяем $x_{n,n-1}$, $x_{n,n-2}, ...,$ $x_{n,1}$:

$$x_{n,i} = -\sum_{k=i+1}^{n} l_{k,i} x_{n,k}, \quad i = n-1, n-2, ..., 1. \tag{3}$$

Далее находим $x_{n-1,n-1}$ из предпоследнего уравнения предпоследней строки (диагональное уравнение) по формуле:

$$x_{i,i} = \frac{1}{u_{i,i}}(1 - \sum_{k=i+1}^{n} u_{i,k} x_{k,i}) \tag{4}$$

при i=n-1. Поднимаясь вверх по столбцу, от этого диагонального уравнения найдем $x_{i,n-1}$ при i=n-2,...,1 по формуле (2), в которой сделаем замену $n \to n-1$ везде, кроме верхнего предела в сумме, а, двигаясь по горизонтали к первому уравнению предпоследней строки, найдем $x_{n-1,i}$ по формуле (3), в которой сделаем замену $n \to n-1$, везде, кроме верхнего предела в суме, i=n-2,...,1. И так далее, определяя элементы x_{ii} из диагональных уравнений, чередуя уравнения части вертикальных столбцов от диагональных уравнений и уравнения части горизонтальных строк от диагональный уравнений, получим все неизвестные.

В результате получаем следующие вычислительные формулы для каждого значения $m = n, n-1, .., 1$:

$$x_{m,m} = \frac{1}{u_{m,m}}(1 - \sum_{k=m+1}^{n} u_{m,k} x_{k,m}),$$

$$x_{i,m} = -\frac{1}{u_{i,i}} \sum_{k=i+1}^{n} u_{i,k} x_{k,m}, \quad i = m-1, ..., 1,$$

$$x_{m,i} = -\sum_{k=i+1}^{n} l_{k,i} x_{m,k}, \quad i = m-1, ..., 1.$$

Рассмотрим пример. Возьмем матрицу A из п.1. Для матрицы A уже найдено LU-разложение, то есть известны значения всех элементов $l_{i,j}$, $u_{i,j}$ матриц L и U, а, значит, выполнены первые 10 команд n.1 или введена команда *LUdecomposition* и выполнена команда $LUdecomposition[a]$.

Введем обратную матрицы с неизвестными пока элементами:

In[11]:= $\mathbf{X = Table[x_{i,j}, \{i, 1, n\}, \{j, 1, n\}]}$;

Положим m=n и выполним следующие три команды команды

In[12]:=

$$\mathbf{x_{m,m} = \frac{1}{u_{m,m}}(1 - \sum_{k=m+1}^{n} u_{m,k} x_{k,m})}$$

$$\mathbf{Do[x_{i,m} = -\frac{1}{u_{i,i}} \sum_{k=i+1}^{n} u_{i,k} x_{k,m}, \{i, m-1, 1-1\}]}$$

$$\mathbf{Do[x_{m,i} = -\sum_{k=i+1}^{n} l_{k,i} x_{m,k}, \{i, m-1, 1, -1\}]}$$

Тем самым выполним команды (1), (2), (3) при m=n. Отметим, что в первой команде при $m = n$ сумма $\sum_{k=n+1}^{n} u_{n,k} x_{k,n} = 0$ и, следовательно, первая команда совпадает с (1) при m=n и с командой (4) при остальных значениях m. Теперь выполним эти три команды, полагая m=n-1,n-2,n-3,...,1. В результате определим все элементы обратной матрицы X.

Сделаем проверку:

In[12]:= **A.X**

В результате должны получить единичную матрицу.

Применим команду цикла для более компактной записи программы. Приведем две команды, необходимые для обращения матрицы: команду $LUdecomposition[a]$, позволяющую получить LU-разложение матрицы a, и команду $ssd[a]$, возвращающую обратную матрицу a^{-1}.

In[1]:=
```
LUdecomposition[a_] :=
    Block[{n = Length[a]},
      U = Table[If[i ≤ j, u_{i,j}, 0], {i, 1, n}, {j, 1, n}];
      L = Table[If[i ≥ j, If[i == j, 1, l_{i,j}], 0], {i, 1, n}, {j, 1, n}];
      Do[u_{1,j} = a[[1, j]], {j, 1, n}];
      Do[l_{i,1} = a[[i, 1]]/u_{1,1}, {i, 2, n}];
      Do[Do[u_{m,i} = a[[m, i]] - Sum_{k=1}^{m-1} l_{m,k} u_{k,i}, {i, m, n}];
      Do[l_{j,m} = 1/u_{m,m}(a[[j, m]] - Sum_{k=1}^{m-1} l_{j,k} u_{k,m}), {j, m + 1, n}], {m, 2, n}];
  ssd[a_] :=
    Block[{n = Length[a]},
      X = Table[x_{i,j}, {i, 1, n}, {j, 1, n}];
      LUdecomposition[a];
      Do[x_{m,m} = 1/u_{m,m}(1 - Sum_{k=m+1}^{n} u_{m,k} x_{k,m});
        Do[x_{i,m} = -1/u_{i,i} Sum_{k=i+1}^{n} u_{i,k} x_{k,m}, {i, m - 1, 1, -1}],
        Do[x_{m,i} = -Sum_{k=i+1}^{n} l_{k,i} x_{m,k}, {i, m - 1, 1, -1}];
      {m, n, 1, -1}];
    X]
```

Перед обращением матрицы следует ввести данные две команды, затем ввести матрицу, например, матрицу A из n.1 и выполнить команду

In[2]:= **ssd[A]**

Проверка:

In[3]:= **ssd[A].A**

§5 Метод квадратных корней Холецкого

Методом квадратных корней (схема Холецкого) решаются системы линейных алгебраических уравнений с симметричной матрицей. Учет симметричности матрицы сокращает число операций при нахождении решения системы приблизительно вдвое по отношению к методу LU-разложения.

Рассмотрим квадратную матрицу $A = (a_{ij})$, например, пятого порядка, n=5. Представим матрицу A в виде произведения: $A = U^\top \cdot U$, где

$$U^\top = \begin{pmatrix} u_{11} & 0 & 0 & 0 & 0 \\ u_{21} & u_{22} & 0 & 0 & 0 \\ u_{31} & u_{32} & u_{33} & 0 & 0 \\ u_{41} & u_{42} & u_{43} & u_{44} & 0 \\ u_{51} & u_{52} & u_{53} & u_{54} & u_{55} \end{pmatrix}, \qquad U = \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} & u_{15} \\ 0 & u_{22} & u_{23} & u_{24} & u_{25} \\ 0 & 0 & u_{33} & u_{34} & u_{35} \\ 0 & 0 & 0 & u_{44} & u_{45} \\ 0 & 0 & 0 & 0 & u_{55} \end{pmatrix}.$$

Перемножив матрицы $U^\top$ и U и, приравняв произведение к матрице A, получим n(n+1)/2 уравнений

$$\begin{array}{ll} u_{11}^2 = a_{11}, & u_{12}u_{11} = a_{12}, \\ u_{12}^2 + u_{22}^2 = a_{22}, & u_{12}u_{13} + u_{22}u_{23} = a_{23}, \\ u_{13}^2 + u_{23}^2 + u_{33}^2 = a_{33}, & u_{13}u_{14} + u_{23}u_{24} + u_{33}u_{34} = a_{34}, \\ u_{14}^2 + u_{24}^2 + u_{34}^2 + u_{44}^2 = a_{44}, & u_{14}u_{15} + u_{24}u_{25} + u_{34}u_{35} + u_{44}u_{45} = a_{45}, \\ u_{15}^2 + u_{25}^2 + u_{35}^2 + u_{45}^2 + u_{55}^2 = a_{55}, & \end{array}$$

$$\begin{array}{lll} u_{13}u_{11} = a_{13}, & u_{14}u_{11} = a_{14}, & u_{15}u_{11} = a_{15}, \\ u_{12}u_{14} + u_{22}u_{24} = a_{24}, & u_{12}u_{15} + u_{22}u_{25} = a_{25}, & \\ u_{13}u_{15} + u_{23}u_{25} + u_{33}u_{35} = a_{35}. & & \end{array}$$

Из первого уравнения первой строки находим

$$u_{11} = \sqrt{a_{11}}, \tag{1}$$

затем из остальных уравнений первой строки:

$$u_{1j} = \frac{a_{1j}}{u_{11}}, \quad j = 2, ..., n. \tag{2}$$

Из первого уравнения второй строки находим

$$u_{22} = \sqrt{a_{22} - u_{12}^2},$$

затем из остальных уравнений второй строки:

$$u_{2j} = \frac{a_{2j} - u_{12}u_{1j}}{u_{22}}, \quad j = 3, ..., n.$$

и так далее. Из последней строки находим ($n = 5$)

$$u_{nn} = \sqrt{a_{nn} - \sum_{k=1}^{n-1} u_{kn}^2}.$$

Таким образом, элементы матрицы U вычисляются по формулам (1), (2) и следующим формулам:

$$u_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} u_{ki}^2}, \quad i = 2, 3, ..., n, \tag{3}$$

$$u_{ij} = \frac{1}{u_{ii}}(a_{ij} - \sum_{k=1}^{i-1} u_{ki}u_{kj}), \quad j = i+1, ..., n, \quad i = 2, 3, ..., n, \tag{4}$$

$$u_{ij} = 0, \quad i < j.$$

Известно, что для положительно определенных симметричных матриц A матрица U - действительная. Таким же свойством обладает матрица A, диагональные элементы у которой положительны и обладают свойством преобладания.

Предположим, что матрица A системы линейных алгебраических уравнений $A.x = b$ допускает разложение: $A = U^\top.U$, тогда система примет вид $U^\top.U.x = b$. Теперь, для получения решения x системы $A.x = b$ необходимо последовательно решить системы $U^\top.y = b$ и $U.x = y$. Так как матрицы этих систем уже имеют треугольный вид, то их решение находится аналогично обратному ходу метода Гаусса.

Отметим положительные стороны метода. Из системы уравнений видим, что $u_{ij} \leqslant \sqrt{a_{ij}}$, то есть элементы матрицы U ограничены.

Такая схема решения самая быстродействующая и имеет хорошую численную устойчивость.

Сгенерируем задание для этого метода. Пусть размерность системы равна

In[1]:= $\mathbf{n = 5}$;

Введем матрицу системы, диагональные элементы которой положительны и обладают свойством преобладания:

In[2]:= $\mathbf{a = Table[If[i == j, Random[Integer, \{n, n + 5\}] + Random[], Random[]], \{i, 1, n\}, \{j, 1, n\}]}$;

и сделаем ее симметричной

In[3]:= $\mathbf{Table[a[[j, i]] = a[[i, j]], \{i, 1, n\}, \{j, i, n\}]}$;

Введем матрицу U:

In[4]:= $\mathbf{U = Table[If[i <= j, u_{i,j}, 0], \{i, 1, n\}, \{j, 1, n\}]}$;

По формуле (1) получим:

In[5]:= $\mathbf{u_{1,1} = Sqrt[a[[1, 1]]]}$;

применяем формулу (2):

In[6]:= $\mathbf{Do[u_{1,i} = \frac{a[[i, 1]]}{u_{1,1}}, \{i, 2, n\}]}$

Составляем циклы для вычисления остальных элементов матрицы U, применяя формулы (3) и (4):

In[7]:= **Do[**

$$\mathbf{u_{i,i}} = \mathbf{Sqrt}[\mathbf{a}[[\mathbf{i},\mathbf{i}]] - \sum_{\mathbf{k=1}}^{\mathbf{i-1}} \mathbf{u_{k,i}^2}];$$
$$\mathbf{Do}[\mathbf{u_{i,j}} = \frac{\mathbf{1}}{\mathbf{u_{i,i}}}(\mathbf{a}[[\mathbf{i},\mathbf{j}]] - \sum_{\mathbf{k=1}}^{\mathbf{i-1}} \mathbf{u_{k,j}u_{k,i}}),$$
$$\{\mathbf{j},\mathbf{i}+\mathbf{1},\mathbf{n}\}],$$
$$\{\mathbf{i},\mathbf{2},\mathbf{n}\}]$$

Для проверки вычислений можно ввести команды

In[8]:= **U**

In[9]:= **Transpose[U].U**

и сравнить результат действия второй команды с матрицей a.

Применим полученное разложение матрицы a для решения системы $a.x = b$.

Введем столбец свободных членов системы уравнений:

In[11]:= $\mathbf{b} = \{\mathbf{1},\mathbf{3},\mathbf{2},\mathbf{5},\mathbf{4}\};$

Обозначим

In[12]:= $\mathbf{L} = \mathbf{Transpose}[\mathbf{U}]$

Теперь находим решение системы $U^{\top}.y = b$:

In[13]:= $\mathbf{y} = \mathbf{Table}[\mathbf{y_i} = (\mathbf{b}[[\mathbf{i}]] - \sum_{\mathbf{j=1}}^{\mathbf{n}} \mathbf{If}[\mathbf{i} == \mathbf{j},\mathbf{0},\mathbf{L}[[\mathbf{i},\mathbf{j}]]]\mathbf{y_j})/\mathbf{L}[[\mathbf{i},\mathbf{i}]],$
$\{\mathbf{i},\mathbf{1},\mathbf{n}\}]$

Out[13]= {0.327354,1.04736,0.54966,1.62635,1.20448}

Решение системы $U.x = y$:

In[14]:= $\mathbf{x} = \mathbf{Table}[\mathbf{x_i} = (\mathbf{y}[[\mathbf{i}]] - \sum_{\mathbf{j=i}}^{\mathbf{n}} \mathbf{If}[\mathbf{i} == \mathbf{j},\mathbf{0},\mathbf{U}[[\mathbf{i},\mathbf{j}]]]\mathbf{x_j})/\mathbf{U}[[\mathbf{i},\mathbf{i}]],$
$\{\mathbf{i},\mathbf{n},\mathbf{1},-\mathbf{1}\}]];$

и получаем ответ исходной систему уравнений:

In[15]:= $\mathbf{x} = \mathbf{Reverse}[\mathbf{x}]//\mathbf{N}$

Out[15]= {0.0598427,0.273293,0.156163,0.537013,0.431237}

Для проверки вычислим вектор невязки:

In[15]:= $\mathbf{b} - \mathbf{a.x}//\mathbf{Chop}$

Out[15]= {0,0,0,0,0,}

§6 Матрицы вращения и решение линейных систем

Метод Гаусса не универсальный метод, в некоторых случаях его нельзя применять. Например, рассмотрим систему с такой расширенной матрицей

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

и приведем ее к треугольному виду

$$A \to \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & -1 & 1 & 2 \\ 0 & -1 & -1 & 2 \end{pmatrix} \to \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & -1 & 4 \end{pmatrix} \to \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 8 \end{pmatrix}.$$

Если у нас матрица порядка n, то будет рост последнего элемента последней строки порядка 2^{n-1}, что, при больших n, может привести к переполнению при решении задачи на компьютере или к сильному влиянию погрешности округления.

Рассмотрим методы, разработанные для такого случая. Рассмотрим решение систем линейных алгебраических уравнений с помощью матриц вращения. Дадим сначала определение матрицы вращения и рассмотрим ее свойства.

Элементарной матрицей вращения T_{ij} называется единичная матрица E, в которой изменены четыре элемента: вместо единиц в позициях (i,i) и (j,j) поставлено число c, в позиции (i,j) поставлено число s, а в позиции (j,i) поставлено $-s$:

$$T_{ij} = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & c & \dots & s & & \\ & & & \ddots & & & \\ & & -s & \dots & c & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix},$$

при этом числа c и s удовлетворяют соотношению: $c^2 + s^2 = 1$.

Свойства матрицы вращения:

Пусть число строк матрица $A = (a_{ij})$ совпадает с размерностью матрицы вращения T_{ij}.

1. Матрица T_{ij} ортогональная: $T_{ij}.T_{ij}^\top = T_{ij}^\top.T_{ij} = E$ и $\det T_{ij} = 1$.

2. При умножении матрицы A слева на матрицу вращения $T_{ij}.A$ изменяются только i и j строки матрицы A. Если $B = T_{ij}.A$ и $B = (b_{ij})$, то

$$b_{im} = ca_{im} + sa_{jm}, \quad b_{jm} = -sa_{im} + ca_{jm}, \ \ m = 1, ..., n, \tag{1}$$

а при $k \neq i, j$

$$b_{km} = a_{km}, \ \ m = 1, ..., n.$$

При умножении справа $A.T_{ij}$ (при согласованных размерностях) изменяются только i и j столбцы матрицы A. Если $B = T_{ij}.A$ и $B = (b_{ij})$, то

$$b_{mi} = ca_{mi} + sa_{mj}, \quad b_{mj} = -sa_{mi} + ca_{mj}, \ \ m = 1, ..., n, \tag{2}$$

а при $k \neq i, j$

$$b_{mk} = a_{mk}, \ \ m = 1, ..., n.$$

Отметим, что преобразование вида (1) над расширенной матрицей системы линейных алгебраических уравнений приводит к расширенной матрице эквивалентной системы.

3. Если

$$c = \frac{a_{ii}}{\sqrt{a_{ii}^2 + a_{ji}^2}}, \qquad s = \frac{a_{ji}}{\sqrt{a_{ii}^2 + a_{ji}^2}},$$

то в матрице $T_{ij}.A$ в позиции (j,i) будет стоять ноль.

4. Суммы квадратов элементов соответствующих столбцов матриц $T_{ij}.A$ и A равны.

Действительно, из (2) получаем

$$\sum_{m=1}^{n} b_{mi}^2 = \sum_{\substack{m=1\\m\neq i,j}}^{n} a_{mi}^2 + b_{ii}^2 + b_{ji}^2 = \sum_{\substack{m=1\\m\neq i,j}}^{n} a_{mi}^2 + (ca_{ii} + sa_{ji})^2 + (-sa_{ii} + ca_{ji})^2 =$$

$$\sum_{\substack{m=1\\m\neq i,j}}^{n} a_{mi}^2 + a_{ii}^2(c^2+s^2) + a_{ji}^2(c^2+s^2) = \sum_{\substack{m=1\\m\neq i,j}}^{n} a_{mi}^2 + a_{ii}^2 + a_{ji}^2 = \sum_{m=1}^{n} a_{mi}^2.$$

Отсюда следует, что при неоднократных умножениях вида $T_{ij}.A$ элементы матрицы произведения не могут неограниченно возрастать.

Рассмотрим систему линейных алгебраических уравнений с расширенной матрицей A. Приведем матрицу системы к треугольному виду, применяя последовательно свойство 3.

Для получения нулей в позициях $(2,1)$, $(3,1)$, ..., $(n,1)$ нужно умножить матрицу A слева на матрицы вращения $T_{1,2}$, $T_{1,3}$, ..., $T_{1,n}$:

$$A^{(1)} = T_{1,n}.T_{1,n-1}...T_{1,2}.A$$

При этом, здесь и далее в подобном случае, коэффициенты матрицы $T_{1,k}$ вычисляются по элементам матрицы $T_{1,k-1}...T_{1,2}.A$ при всех $k = 3,4,...,n-1$.

Ясно, что системы с расширенными матрицами A и $A^{(1)}$ эквивалентны. Умножим матрицу $A^{(1)}$ слева на матрицы вращения $T_{2,3}$, затем полученную матрицу умножим на $T_{2,4}$, и так далее, последнее умножение производим на матрицу $T_{2,n}$. В результате получим матрицу $A^{(2)}$:

$$A^{(2)} = T_{2,n}.T_{2,n-1}...T_{2,3}.A^{(1)},$$

у которой поддиагональные элементы первых двух столбцов будут нулевыми. Продолжая аналогичные умножения, на $n-1$ шаге получим треугольную матрицу

$$A^{(n-1)} = T_{n-1,n}.A^{(n-2)}.$$

Система с расширенной матрицей $A^{(n-1)}$ равносильна системе с расширенной матрицей A. Для получения решения системы с треугольной матрицей $A^{(n-1)}$ осуществляется обратный ход такой же, как и в методе Гаусса.

Рассмотрим пример обнуления одного элемента расширенной матрицы системы уравнений при помощи матрицы вращения.

Зададим размерность расширенной матрицы:

In[1]:= $\mathbf{n = 5}$;

Генерируем расширенную матрицу системы:

In[2]:= $\mathbf{a = Table[Random[], \{i, 1, n\}, \{j, 1, n+1\}]}$;

Обнулим элемент матрицы a в позиции (2,1). Положим

In[3]:= $\mathbf{i = 1}$;

$\mathbf{j = 2}$;

и вычислим элементы c и s матрицы вращения в соответствии со свойством 3:

In[4]:= **c = a[[i, i]]/Sqrt[a[[i, i]]^2 + a[[j, i]]^2]//N;**
s = a[[j, i]]/Sqrt[a[[i, i]]^2 + a[[j, i]]^2]//N;

При помощи преобразования $a = T_{12}.a$ в матрице a изменяем строки i, j по формулам (1):

In[5]:= **ap = c a[[i]] + s a[[j]]//N;** (* Запоминаем новую i-ую строку, не меняя старой. *)
a[[j]] = −s a[[i]] + c a[[j]]//N; (* Изменяем j строку.*)
a[[i]] = ap; (* Изменяем i строку.*)

В новой матрице a на месте элемента (j, i) стоит ноль:

In[6]:= **a//Chop**

Рассмотрим теперь пример приведения матрицы a к треугольному виду. Для этого введем циклы по $i = 1, ..., n$ и $j = 1, ..., i$, обнуляющие все поддиагональные элементы матрицы a. Перед следующей командой надо ввести матрицу и ее размерность.

In[1]:=
```
Do[
 Do[
  c = a[[i, i]]/Sqrt[a[[i, i]]^2 + a[[j, i]]^2]//N;
  s = a[[j, i]]/Sqrt[a[[i, i]]^2 + a[[j, i]]^2]//N;
  ap = c a[[i]] + s a[[j]]//N;
  a[[j]] = −s a[[i]] + c a[[j]]//N;
  a[[i]] = ap,
 {j, i, n}],
{i, 1, n}]
```

В результате получим:

In[2]:= **a//Chop**

Наконец, введем следующую команду, решающую систему уравнений при помощи матриц вращения.

In[1]:=
```
vrashenie[a1_] := Block[{a = a1, n = Length[a1]},
Do[
Do[
c = a[[j, j]]/Sqrt[a[[j, j]]^2 + a[[i, j]]^2]//N;
s = a[[i, j]]/Sqrt[a[[j, j]]^2 + a[[i, j]]^2]//N;
ap = c a[[j]] + s a[[i]]//N;
a[[i]] = −s a[[j]] + c a[[i]]//N;
a[[j]] = ap;,
{i, j, n}],
{j, 1, n}];
(* Обратный ход, как в методе Гаусса *)
x = Table[x_i = (a[[i, n + 1]] − Sum_{j=i}^{n} If[i == j, 0, a[[i, j]]]x_j)/a[[i, i]],
 {i, n, 1, −1}];
x = Reverse[x]]
```

Для применения команды нужно ввести расширенную матрицу системы a и выполнить команду

In[2]:= **vrashenie[a]**

§7 Матрицы отражения

Рассмотрим решение систем линейных алгебраических уравнений с помощью матриц отражения. Дадим сначала определение матрицы отражения и рассмотрим ее свойства.

Матрицей отражения называется матрица

$$U = E - 2w.w^{\top}, \tag{1}$$

где w есть n-вектор-столбец единичной длины: $|w| = 1$, а E - единичная матрица порядка n.

Свойства матрицы отражения:

1. U - ортогональная матрица. Действительно, $U.U^{\top} = (E-2w.w^{\top}).\ (E-2w.w^{\top})^{\top} = (E-2w.w^{\top}).(E-2(w.w^{\top})^{\top}) = (E-2w.w^{\top}).(E-2w.w^{\top}) = E-4w.w^{\top}+4w.(w^{\top}.w).w^{\top} = E$ так как

$$w^{\top}.w = (w_1, ..., w_n).\begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix} = (w_1w_1 + ... + w_nw_n) = (w, w) = 1$$

Здесь (x, y) - скалярное произведение векторов.

2. Если $z \perp w$, то $U.z = z$. Действительно, $U.z = (E - 2w.w^{\top}).z = z - 2w.w^{\top}.z = z - 2w(w, z) = z$, так как векторы z и w перпендикулярны, следовательно скалярное произведение $(w, z) = 0$.

3. $U.w = -w$. Действительно, $U.w = (E - 2w.w^{\top}).w = w - 2w.w^{\top}.w = w - 2w(w, w) = -w$.

4. Для вектора s и единичного вектора e существует вектор w, такой, что матрица отражения, построенная по этому вектору, обладает следующим свойством: $U.s = \alpha e$, при некотором α.

Действительно, достаточно взять $w = (s-\alpha e)/\rho$, $\alpha = -sign(s, e)|s|$, $\rho = |s-\alpha e| = \sqrt{2\alpha^2 - 2\alpha(s, e)}$.

Тогда $U.s = s - 2w.w^{\top}.s = s - 2(w, s)w = s - \frac{1}{\rho}(2\alpha^2 - 2\alpha(e, s))w = s - \frac{1}{\rho}\rho^2 w = s - \rho w = s - s + \alpha e = \alpha e$.

5. $det(U) = -1$. Пусть $e_1, ..., e_n$ - ортонормированный базис, такой, что векторы $e_1, ..., e_{n-1}$ перпендикулярны w, а вектор $e_n = w$. Тогда $U.e_i = e_i$, $i = 1, ..., n-1$, $U.e_n = -e_n$. Пусть матрица $D = (e_1, ..., e_n)$, где координаты векторов e_i являются соответствующими столбцами. Тогда $U.D = -D$, $det(U)det(D) = -det(D)$ и $det(U) = -1$.

n.1 Решение линейных систем с помощью матрицы отражения

Рассмотрим систему линейных алгебраических уравнений с расширенной матрицей A порядка $n \times (n + 1)$. Для получения нулей в матрице A в позициях $(2, 1)$, $(3, 1), ...,$ $(n, 1)$ возьмем первый столбец s_1 матрицы A, и пусть $e_1 = (1, 0, ..., 0)^{\top}$ - n вектор-столбец. Построим по векторам s_1 и e_1 вектор w_1 и матрицу отражения $U_1 = E - 2w_1.w_1^{\top}$. Теперь обнуляем элементы матрицы $A_0 = A$ в указанных позициях: $A_1 = U_1.A$. В силу свойства 4 матрицы отражения первый столбец матрицы A_1

будет иметь вид $U_1.s_1 = \alpha_1 e_1$, при некотором α_1. Вектор $\alpha_1 e_1 = (\alpha_1, 0, ..., 0)^\top$, поэтому матрицу A_1 в блочном виде можно записать так

$$A_1 = \begin{pmatrix} \alpha_1 & | & V_1 \\ -- & -- & -- \\ 0 & | & B_1 \end{pmatrix},$$

где V_1 - $(n-1)$-вектор, 0 - $(n-1)$-вектор столбец состоящий из нулей, B_1 - квадратная матрица порядка $n-1$.

Далее, берем матрицу B_1, имеющую размерности $n-1$, и обнуляем все элементы первого столбца этой матрицы, начиная со второго, повторяя предыдущую процедуру, и так далее.

На $i-1$ шаге получим матрицу системы, которую в блочном виде можно записать так

$$A_{i-1} = \begin{pmatrix} \alpha_{i-1} & | & V_{i-1} \\ -- & -- & -- \\ 0 & | & B_{i-1} \end{pmatrix}, \tag{2}$$

где B_{i-1} - матрица порядка $(n-i+1) \times (n-i+1)$, 0 - нулевая матрица порядка $(n-i+1) \times (i-1)$, V_{i-1} - матрица порядка $(i-1) \times (n-i+1)$, а

$$\alpha_{i-1} = \begin{pmatrix} \alpha_1 & * & * & ...* \\ 0 & \alpha_2 & * & ...* \\ \vdots & \vdots & \vdots & * \\ 0 & 0 & ... & \alpha_{i-1} \end{pmatrix},$$

где "*"-ки обозначают некоторые, вообще говоря, ненулевые элементы - это начальные элементы строк матриц $V_1, ..., V_{i-2}$.

Считаем, что $A_0 = A$, $B_0 = A$.

После $n-1$ шага матрица (2) примет треугольный вид, под главной диагональю матрицы (2) будут стоять нули.

Рассмотрим еще один способ применения матрицы отражения для приведения матрицы системы к треугольному виду, не используя переход к матрицам меньших размерностей. Для этого рассмотрим i шаг предыдущего метода. Пусть $s' = (a_{i,i}, a_{i+1,i}, ..., a_{n,i})^\top$ первый столбец матрицы B_{i-1} в формуле (2), $e' = (1, 0, ..., 0)^\top$ - $(n-i)$ - вектор столбец. При $i = 1$, s' - первый столбец матрицы A. По s' и e' вычислим α как в свойстве (4). Для обнуления элементов первого столбца матрицы B_{i-1} (кроме первого элемента) возьмем вектор $w' = (s' - \alpha e')/||s' - \alpha e'|| = (a_{i,i} - \alpha, a_{i+1,i}, ..., a_{n,i})^\top / ||s' - \alpha e||$ и пусть $U' = E'' - 2w'.w'^\top$ - матрица отражения, построенная по вектору w', E''-$(n-i+1)$-мерная единичная матрица.

Произведение $U'.B_{i-1}$, обнуляющее элементы первого столбца матрицы B_{i-1} в (2) начиная со второго элемента, получим как следствие в результате перемножения матриц порядка n. Пусть

$$s = (\underbrace{0, ..., 0}_{i-1}, a_{i,i}, a_{i+1,i}, ..., a_{n,i})^\top, \tag{3}$$

$$e = (\underbrace{0, ..., 0}_{i-1}, 1, 0, ..., 0)^\top. \tag{4}$$

Рассмотри вектор

$$w = (s - \alpha e)/||s - \alpha e|| = (\underbrace{0, ..., 0}_{i-1}, a_{i,i} - \alpha, a_{i+1,i}, ..., a_{n,i})^\top / ||s - \alpha e||$$

или в блочном виде $w = (0|w')^\top$. По вектору w построим матрицу отражения

$$U = E - 2w.w^\top = \begin{pmatrix} E' & | & 0 \\ - & - & - \\ 0 & | & E'' \end{pmatrix} - 2 \begin{pmatrix} 0 \\ - \\ w' \end{pmatrix} .(0|w'^\top) = \begin{pmatrix} E' & | & 0 \\ - & - & - \\ 0 & | & E'' \end{pmatrix} -$$
$$2 \begin{pmatrix} 0 & | & 0 \\ -- & -- & -- \\ 0 & | & w'.w'^\top \end{pmatrix} = \begin{pmatrix} E' & | & 0 \\ -- & -- & -- \\ 0 & | & E'' - 2w'.w'^\top \end{pmatrix} = \begin{pmatrix} E' & | & 0 \\ -- & -- & -- \\ 0 & | & U' \end{pmatrix}.$$

Здесь E'-$(i-1)$-мерная единичная матрица. Теперь нужное преобразование матрицы B_{i-1} можно получить, перемножая n-матрицы

$$U.A_{i-1} = \begin{pmatrix} E' & | & 0 \\ -- & -- & -- \\ 0 & | & U' \end{pmatrix} . \begin{pmatrix} \alpha_{i-1} & | & V_{i-1} \\ -- & -- & -- \\ 0 & | & B_{i-1} \end{pmatrix} =$$
$$\begin{pmatrix} \alpha_{i-1} & | & V_{i-1} \\ -- & -- & -- \\ 0 & | & U'.B_{i-1} \end{pmatrix}. \qquad (5)$$

Последовательность умножений $A_i = U_i.A_{i-1}$, при $i = 1, 2, ..., n-1$, приводит к левой треугольной матрице, если начальная матрица $A_0 = A$, а матрица U_i формируется по матрице A_{i-1} так, как указано выше.

Рассмотрим конкретные примеры применения двух вариантов метода отражения к решению линейных систем. Начнем со второго метода, приводящего матрицу системы к треугольному виду с помощью перемножения матриц порядка n.

n.2 Метод отражения с преобразованием расширенной матрицы

Введем команду вычисления нормы вектора или матрицы:

In[1]:= **nr[a_] := Sqrt[(a//Flatten).(a//Flatten)]**

Пусть размерность расширенной матрицы $n \times (n+1)$, где

In[2]:= **n = 5;**

Сгенерируем расширенную матрицу системы:

In[3]:= **a = Table[Random[], {i, 1, n}, {j, 1, n + 1}];**

Обнулим элементы первого столбца матрицы a начиная со второго элемента. Для этого в формуле (3) положим $i = 1$ для матрицы a, то есть введем первый столбец матрицы a:

In[4]:= **s = a[[All, 1]];**

Следующий набор команд задает, числа α и ρ, вектор w и матрицу отражения u. Последняя, 10-ая команда реализует умножение (5) и обнуляет отмеченные элементы и переопределяет расширенную матрицу:

In[5]:= **e = {1, 0, 0, 0, 0};**

In[6]:= $\alpha = -\mathbf{Sign}[\mathbf{s.e}]\mathbf{nr}[\mathbf{s}]//\mathbf{N};$
In[7]:= $\rho = \mathbf{nr}[\mathbf{s} - \alpha\mathbf{e}]//\mathbf{N};$
In[8]:= $\mathbf{w} = \mathbf{Transpose}[\{\mathbf{1}/\rho(\mathbf{s} - \alpha\mathbf{e})\}];$
In[9]:= $\mathbf{u} = \mathbf{IdentityMatrix}[\mathbf{n}] - \mathbf{2w.Transpose}[\mathbf{w}];$
In[10]:= $\mathbf{a} = \mathbf{u.a}//\mathbf{Chop}$
Out[10]=

$$\begin{pmatrix} 1.08413 & 0.88103 & 0.22074 & 1.211456 & 0.50698 & 0.63811 \\ 0 & 0.61976 & 0.40316 & 0.63846 & 0.14549 & 0.02060 \\ 0 & -0.14327 & 0.27036 & 0.41775 & 0.26489 & 0.76394 \\ 0 & 0.26112 & 0.04725 & 0.05408 & 0.27774 & 0.17319 \\ 0 & 0.80094 & 0.06492 & 0.16076 & 0.56424 & 0.12959 \end{pmatrix}$$

На втором шаге вводим второй столбец расширенной матрицы и обнуляем у него первый элемент, то есть вводим вектором s в соответствии с формулой (3) при $i = 2$, вектор e по формуле (4). Остальные команды такие же, как и на первом шаге.

In[11]:= $\mathbf{s} = \mathbf{a}[[\mathbf{All}, \mathbf{2}]];$
In[12]:= $\mathbf{s}[[\mathbf{1}]] = \mathbf{0};$
In[13]:= $\mathbf{e} = \{\mathbf{0}, \mathbf{1}, \mathbf{0}, \mathbf{0}, \mathbf{0}\};$
In[14]:= $\alpha = -\mathbf{Sign}[\mathbf{s.e}]\mathbf{nr}[\mathbf{s}]//\mathbf{N};$
In[15]:= $\rho = \mathbf{nr}[\mathbf{s} - \alpha\mathbf{e}]//\mathbf{N};$
In[16]:= $\mathbf{w} = \mathbf{Transpose}[\{\mathbf{1}/\rho(\mathbf{s} - \alpha\mathbf{e})\}];$
In[17]:= $\mathbf{u} = \mathbf{IdentityMatrix}[\mathbf{n}] - \mathbf{2w.Transpose}[\mathbf{w}];$
In[18]:= $\mathbf{a} = \mathbf{u.a}//\mathbf{Chop}$
Out[18]=

$$\begin{pmatrix} 1.08413 & 0.88103 & 0.22074 & 1.211456 & 0.50698 & 0.63811 \\ 0 & 1.05561 & 0.26095 & 0.45351 & 0.54628 & 0.04959 \\ 0 & 0 & 0.32715 & 0.51109 & 0.32405 & 0.76994 \\ 0 & 0 & 0.05626 & 0.11611 & -0.16995 & -0.16226 \\ 0 & 0 & 0.25257 & 0.36127 & -0.23353 & -0.09604 \end{pmatrix}$$

На третьем шаге вводим третий столбец расширенной матрицы, обнуляем у него первые два элемента и повторяем снова все команды, приводящие к обнулению всех поддиагональных элементов третьего столбца расширенной матрицы:

In[19]:= $\mathbf{s} = \mathbf{a}[[\mathbf{All}, \mathbf{3}]];$
In[20]:= $\mathbf{s}[[\mathbf{1}]] = \mathbf{s}[[\mathbf{2}]] = \mathbf{0};$
In[21]:= $\mathbf{e} = \{\mathbf{0}, \mathbf{0}, \mathbf{1}, \mathbf{0}, \mathbf{0}\};$
In[22]:= $\alpha = -\mathbf{Sign}[\mathbf{s.e}]\mathbf{nr}[\mathbf{s}]//\mathbf{N};$
In[23]:= $\rho = \mathbf{nr}[\mathbf{s} - \alpha\mathbf{e}]//\mathbf{N};$
In[24]:= $\mathbf{w} = \mathbf{Transpose}[\{\mathbf{1}/\rho(\mathbf{s} - \alpha\mathbf{e})\}];$
In[25]:= $\mathbf{u} = \mathbf{IdentityMatrix}[\mathbf{n}] - \mathbf{2w.Transpose}[\mathbf{w}];$
In[26]:= $\mathbf{a} = \mathbf{u.a}//\mathbf{Chop}$
Out[26]=

$$\begin{pmatrix} 1.08413 & 0.88103 & 0.22074 & 1.211456 & 0.50698 & 0.63811 \\ 0 & 1.05561 & 0.26095 & 0.45351 & 0.54628 & 0.04959 \\ 0 & 0 & 0.41711 & 0.63527 & 0.08983 & 0.523840 \\ 0 & 0 & 0 & 0.02945 & -0.2012 & -0.26006 \\ 0 & 0 & 0 & -0.02776 & -0.37399 & -0.53509 \end{pmatrix}$$

Аналогично проводим четвертый шаг.

In[27]:= $\mathbf{s} = \mathbf{a}[[\mathbf{All}, \mathbf{4}]];$

```
In[28]:= s[[1]] = s[[2]] = s[[3]] = 0;
In[29]:= e = {0, 0, 0, 1, 0};
In[30]:= α = -Sign[s.e]nr[s]//N;
In[31]:= ρ = nr[s - αe]//N;
In[32]:= w = Transpose[{1/ρ(s - αe)}];
In[33]:= u = IdentityMatrix[n] - 2w.Transpose[w];
In[34]:= a = u.a//Chop
```

Out[34]=

$$\begin{pmatrix} 1.08413 & 0.88103 & 0.22074 & 1.211456 & 0.50698 & 0.63811 \\ 0 & 1.05561 & 0.26095 & 0.45351 & 0.54628 & 0.04959 \\ 0 & 0 & 0.41711 & 0.63527 & 0.08983 & 0.523840 \\ 0 & 0 & 0 & 0.04046 & 0.11007 & 0.17773 \\ 0 & 0 & 0 & 0 & -0.41016 & -0.56777 \end{pmatrix}$$

В результате приведем расширенную матрицу системы к треугольному виду. Обратный ход такой же, как и в методе Гаусса.

Воспользуемся командой цикла для компактной записи команд всех рассмотренных шагов. Вектор e можно задать с помощью вектора-заготовки p, в котором соответствующая нулевая координата заменяется единицей, а для обнуления элементов столбца расширенной матрицы воспользуемся командой *Do*.

```
In[1]:= otrazenie1[a1_] := Block[{n = Length[a1], a = a1},
          p = Table[0, {n}];
          s = a[[All, 1]];(* Первый столбец для первого прохода следующего цикла. *)
          Do[e = p;
          e[[i]] = 1;
          α = -Sign[s.e]nr[s]//N;
          ρ = nr[s - αe]//N;
          w = Transpose[{1/ρ(s - αe)}];
          u = IdentityMatrix[n] - 2w.Transpose[w];
          a = u.a;
    (* Обнуление первых i элементов i + 1 столбца s *)
          s = a[[All, i + 1]];
          Do[s[[k]] = 0, {k, 1, i}],
          {i, 1, n - 1}];
    (* Обратный ход *)
         x = Table[x_i = (a[[i, n + 1]] - ∑_{j=i}^{n} If[i == j, 0, a[[i, j]]]x_j)/a[[i, i]],
          {i, n, 1, -1}];
         x = Reverse[x]]
```

Теперь для получения решения системы достаточно ввести расширенную матрицу системы (с именем a) и выполнить команду

```
In[2]:= otrazenie1[a]
```

n.3 Метод отражения с преобразованием подматриц

Введем команду вычисления нормы вектора или матрицы:

```
In[1]:= nr[a_] := Sqrt[(a//Flatten).(a//Flatten)]
```

зададим число строк расширенной матрицы:

In[2]:= $\mathbf{n = m = 5}$;

Генерируем расширенную матрицу системы:

In[3]:= $\mathbf{a = Table[Random[], \{i, 1, n\}, \{j, 1, n + 1\}]}$;

Проделаем теперь ряд однотипных шагов и приведем расширенную матрицу a к треугольному виду. На каждом шаге будем строить матрицу вида (2), преобразовывая каждый раз подматрицы B_{i-1}.

На первом шаге обнулим элементы первого столбца матрицы начиная со второго так же, как и в первом методе. Для этого выделим первый столбец:

In[4]:= $\mathbf{s = a[[All, 1]]}$;

Следующие команды вводят вектор e, числа α и ρ, вектор w и матрицу отражения u. Команда 11 обнуляет отмеченные элементы и переопределяет расширенную матрицу

In[5]:= $\mathbf{e = Table[0, \{n\}]}$;

In[6]:= $\mathbf{e[[1]] = 1}$;

In[7]:= $\alpha = \mathbf{-Sign[s.e]nr[s]//N}$;

In[8]:= $\rho = \mathbf{nr[s - \alpha e]//N}$;

In[9]:= $\mathbf{w = Transpose[\{1/\rho(s - \alpha e)\}]}$;

In[10]:= $\mathbf{u = IdentityMatrix[n] - 2w.Transpose[w]}$;

In[11]:= $\mathbf{a = u.a//Chop}$

Запомним матрицу a: ее первый столбец и первая строка войдут в итоговую расширенную матрицу треугольного вида.

In[12]:= $\mathbf{b_1 = a//Chop}$

Вырезая из матрицы a первый столбец и первую строку, получим подматрицу (B_1 в формулы (2) при $i = 2$), которую обозначим так же a:

In[13]:= $\mathbf{a = a[[2;;, 2;;]]}$;

На втором шаге уменьшаем размерность n на единицу, что совпадает с числом строк матрицы a ($=B_1$), вырезаем из вектора e последний ноль и повторяем все команды с 7 по 13, полагая при этом $i = 2$ в 22 команде:

In[14]:= $\mathbf{n = n - 1}$;

In[15]:= $\mathbf{e = Drop[e, -1]}$;;

In[16]:= $\mathbf{s = a[[All, 1]]}$;

In[17]:= $\alpha = \mathbf{-Sign[s.e]nr[s]//N}$;

In[18]:= $\rho = \mathbf{nr[s - \alpha e]//N}$;

In[19]:= $\mathbf{w = Transpose[\{1/\rho(s - \alpha e)\}]}$;

In[20]:= $\mathbf{u = IdentityMatrix[n] - 2w.Transpose[w]}$;

In[21]:= $\mathbf{a = u.a//Chop}$

In[22]:= $\mathbf{b_i = a//Chop}$

In[23]:= $\mathbf{a = a[[2;;, 2;;]]}$;

Для выполнения третьего и четвертого шагов выполним команды 14-23 дважды, при i=3,4 в 22 команде. При этом у матрицы a дважды будут вырезаны первые строки и столбцы. В результате выполнения четырех шагов, получим матрицу $b_4 = a$ в виде

Out[34]=

$$\begin{pmatrix} 0.04046 & 0.11007 & 0.17773 \\ 0 & -0.41016 & -0.56777 \end{pmatrix}$$

Для восстановления расширенной матрицы системы возьмем нулевую матрицу исходной размерности:

In[24]:= $\mathbf{c = Table[0, \{i, 1, m\}, \{j, 1, m + 1\}]}$;

и изменим в ней соответствующие элементы.

Заменим последнюю пятую строку матрицы c - на последнюю строку матрицы b_4, добавленную в начале 3 нулями,

In[25]:= $\mathbf{c[[m]] = Join[\{0,0,0\}, b_{m-1}[[2]]];}$

4-ую строку матрицы c - на первую строку матрицы b_4, добавленную в начале 3 нулями

In[26]:= $\mathbf{c[[m-1]] = Join[\{0,0,0\}, b_{m-1}[[1]]];}$

3-ую строку матрицы c - на первую строку матрицы b_3, добавленную в начале 2 нулями

In[27]:= $\mathbf{c[[m-2]] = Join[\{0,0\}, b_{m-2}[[1]]];}$

2-ую строку матрицы c - на первую строку матрицы b_2, добавленную в начале 1 нулем

In[28]:= $\mathbf{c[[m-3]] = Join[\{0\}, b_{m-3}[[1]]];}$

и 1-ую строку матрицы c - на первую строку матрицы b_1,

In[29]:= $\mathbf{c[[1]] = b_1[[1]];}$

В результате получим матрицу системы в треугольном виде.

Отметим, что строки 26-28 можно записать компактно в следующем виде

$\mathbf{Do[c[[m-i]] = Join[Table[0,\{m-i-1\}], b_{m-i}[[1]]], \{i,1,m-1\}];}$

После приведения матрицы системы к треугольному виду осуществляется обратный ход такой же, как и в методе Гаусса.

Объединим все команды в одну команду otrazenie2[a]:

In[1]:=
```
otrazenie2[a1_] := Block[{m = n = Length[a1], a = a1},
    e = Table[0, {n}];
    e[[1]] = 1;
    Do[
    s = a[[All, 1]];
    α = -Sign[s.e]nr[s]//N;
    ρ = nr[s - αe]//N;
    w = Transpose[{1/ρ(s - αe)}];
    u = IdentityMatrix[n] - 2 w.Transpose[w];
    a = u.a//Chop;
(* Запоминаем матрицу a *)
    b_i = a;
(* Вырезаем первые строку и столбец *)
    a = a[[2;;, 2;;]];
    n = n - 1;
    e = Drop[e, -1],
    {i, 1, n - 1}];
(* Восстановление расширенной матрицы *)
    c = Table[0, {i, 1, m}, {j, 1, m + 1}];
    c[[m, m + 1]] = b_{m-1}[[m - 3, m - 2]];
    c[[m, m]] = b_{m-1}[[m - 3, m - 3]];
    Do[c[[m - i]] = Join[Table[0, {m - i - 1}], b_{m-i}[[1]]], {i, 1, m - 1}];
    c[[1]] = b_1[[1]];
(* Обратный ход *)
    x = Table[x_i = (a[[i, n + 1]] - Sum_{j=i}^{n} If[i == j, 0, a[[i, j]]]x_j)/a[[i, i]],
    {i, n, 1, -1}];
```

$\mathbf{x} = \mathbf{Reverse}[\mathbf{x}]]$

§8 Итерационные методы решения систем линейных алгебраических уравнений

Рассмотрим два итерационных метода решения систем линейных алгебраических уравнений. Пусть дана система уравнений, записанная в матричном виде

$$A.x = b. \tag{1}.$$

Приведем эту систему некоторыми равносильными преобразованиями к следующему виду:

$$x = B.x + c, \tag{2}$$

и построим последовательность векторов $\{x^{(k)}\}_{k=1,2,...}$, начиная с некоторого начального приближения $x^{(0)}$, по формуле

$$x^{(k)} = B.x^{(k-1)} + c. \tag{2'}$$

Теорема 1. Последовательность $\{x^{(k)}\}$ сходится к решению ξ системы уравнений (1), если норма матрицы B системы (2) удовлетворяет условию:

$$||B|| \leqslant q < 1 \tag{3}$$

при некотором q. При этом погрешность приближенного решения $x^{(k)}$ допускает следующую оценку:

$$||\xi - x^{(k)}|| < \frac{q}{1-q}||x^{(k)} - x^{(k-1)}||. \tag{4}$$

Доказательство. Пусть $\overline{x}$ - решение однородной системы

$$x = B.x \quad \text{или} \quad (B - E).x = 0, \tag{5}$$

где E - единичная матрица.

Покажем, что $\overline{x} = 0$. Так как $\overline{x} = B.\overline{x}$, то $||\overline{x}|| = ||B.\overline{x}|| \leqslant ||B||\,||\overline{x}||$ или $||\overline{x}||(1 - ||B||) \leqslant 0$. Отсюда и из (3) следует, что $||\overline{x}|| = 0$ или $\overline{x} = 0$. Известно, что тогда неоднородная система (2) имеет единственное решение при любом столбце свободных членов. Пусть ξ - решение системы (2). Имеем

$$\xi = B.\xi + c, \quad x^{(k)} = B.x^{(k-1)} + c. \tag{6}$$

Отсюда

$$||\xi - x^{(k)}|| = ||B.\xi - B.x^{(k-1)}|| = ||B.(\xi - x^{(k-1)})|| \leqslant$$
$$||B||\,||(\xi - x^{(k-1)})|| \leqslant ||B||^2\,||\xi - x^{(k-2)}|| \leqslant ... \leqslant ||B||^k\,||\xi - x^{(0)}||.$$

Поэтому при $k \to \infty$ в силу (3) получаем сходимость последовательности $\{x^{(k)}\}$ к решению системы ξ.

Для оценки погрешности приближенного решения $x^{(k)}$ из (6) запишем:

$$\xi - x^{(k)} = B.(\xi - x^{(k-1)}) = B.\xi - B.x^{(k-1)} + B.x^{(k)} - B.x^{(k)} = B.(\xi - x^{(k)}) + B.(x^{(k)} - x^{(k-1)}).$$

Тогда норма левой части

$$||\xi - x^{(k)}|| \leqslant ||B.(\xi - x^{(k)})|| + ||B.(x^{(k)} - x^{(k-1)})|| \leqslant$$

$$||B||\,||(\xi - x^{(k)})|| + ||B||\,||(x^{(k)} - x^{(k-1)})||.$$

Отсюда получаем

$$||\xi - x^{(k)}|| < \frac{||B||}{1 - ||B||}||x^{(k)} - x^{(k-1)}||$$

или, учитывая (3), получаем оценку (4).

Приведем без доказательства еще одну теорему о сходимости итерационной последовательности.

Теорема 2. Последовательность $\{x^{(k)}\}$ сходится к решению ξ системы уравнений (1), если

а) наибольшее по модулю собственное значение матрицы B меньше единицы;

б) матриц $A = (a_{i,j})$ системы (1) есть матрица с преобладанием диагональных элементов:

$$|a_{i,i}| > \sum_{\substack{j=1,2,\dots \\ j \neq i}} |a_{i,j}| \qquad (\text{или} \quad 2\,|a_{i,i}| > \sum_{j=1}^{n} |a_{i,j}|)$$

для всех i.

Построение члена такой последовательность, являющегося приближенным значением вектора решения данной системы с заданной точностью $\varepsilon > 0$, составляет суть метода итераций.

Методы Якоби и Зейделя характеризуются способами приведения системы (1) к виду (2).

Метод Якоби. В методе Якоби предполагается, что элементы главной диагонали матрицы системы не равны нулю. Матрицу системы A представляют как сумму $A = l + d + r$, где l - левая треугольная матрица, то есть все элементы над главной диагональю и элементы главной диагонали равны нулю, r - правая треугольная матрица, то есть все элементы под главной диагональю и элементы главной диагонали равны нулю, а d - диагональная матрица, то есть не нулевые только элементы главной диагонали. Тогда систему (1) можно переписать так: $(l + d + r).x = b$, или $d.x = -(l + r).x + b$, или

$$x = -d^{-1}.(l + r).x + d^{-1}.b. \tag{7}$$

Из представления (7) получаем итерационную формулу (2′) метода Якоби:

$$x^{(k)} = -d^{-1}.(l + r).x^{(k-1)} + d^{-1}.b \tag{7′}$$

и условие сходимости итерационной последовательности по формуле (3):

$$|| - d^{-1}.(l + r)|| \leqslant q < 1.$$

В координатном виде формула (7′) для случая системы с 5-ю уравнениями и 5-ю неизвестными записывается так:

$$\begin{cases} x_1^{(k)} = s_{1,1}x_1^{(k-1)} + s_{1,2}x_2^{(k-1)} + s_{1,3}x_3^{(k-1)} + s_{1,4}x_4^{(k-1)} + s_{1,5}x_5^{(k-1)} + u_1, \\ x_2^{(k)} = s_{2,1}x_1^{(k-1)} + s_{2,2}x_2^{(k-1)} + s_{2,3}x_3^{(k-1)} + s_{2,4}x_4^{(k-1)} + s_{2,5}x_5^{(k-1)} + u_2, \\ x_3^{(k)} = s_{3,1}x_1^{(k-1)} + s_{3,2}x_2^{(k-1)} + s_{3,3}x_3^{(k-1)} + s_{3,4}x_4^{(k-1)} + s_{3,5}x_5^{(k-1)} + u_3, \\ x_4^{(k)} = s_{4,1}x_1^{(k-1)} + s_{4,2}x_2^{(k-1)} + s_{4,3}x_3^{(k-1)} + s_{4,4}x_4^{(k-1)} + s_{4,5}x_5^{(k-1)} + u_4, \\ x_5^{(k)} = s_{5,1}x_1^{(k-1)} + s_{5,2}x_2^{(k-1)} + s_{5,3}x_3^{(k-1)} + s_{5,4}x_4^{(k-1)} + s_{5,5}x_5^{(k-1)} + u_5, \end{cases}$$

здесь $x_1^{(k)}, ..., x_5^{(k)}$ координаты k-го вектора итерации, а $s_{i,j}$ и u_i элементы соответственно матриц

$$B = -d^{-1}.(l + r) \quad \text{и} \quad c = d^{-1}.b \tag{8}$$

Отметим, что координаты k-го вектора итерации $x^{(k)}$ выражаются только через координаты $(k-1)$-го вектора $x^{(k-1)}$.

Метод Зейделя. Метод Зейделя является усилением метода Якоби и характеризуется тем, что при вычислении i-ой координаты k-го вектора $x^{(k)}$ итерационной последовательсти учитывается то, что координаты $x_j^{(k)}$ уже найдены для j<i. Из свойства компьютерной программы выполнять команды в порядке их следования, вытекает, что итерационную формулу метода Зейделя можно реализовать программно в следующем виде, убрав предварительно номера векторов:

$$\begin{cases} x_1^{(k)} = s_{1,1}x_1^{(k-1)} + s_{1,2}x_2^{(k-1)} + s_{1,3}x_3^{(k-1)} + s_{1,4}x_4^{(k-1)} + s_{1,5}x_5^{(k-1)} + u_1, \\ x_2^{(k)} = s_{2,1}x_1^{(k)} + s_{2,2}x_2^{(k-1)} + s_{2,3}x_3^{(k-1)} + s_{2,4}x_4^{(k-1)} + s_{2,5}x_5^{(k-1)} + u_2, \\ x_3^{(k)} = s_{3,1}x_1^{(k)} + s_{3,2}x_2^{(k)} + s_{3,3}x_3^{(k-1)} + s_{3,4}x_4^{(k-1)} + s_{3,5}x_5^{(k-1)} + u_3, \\ x_4^{(k)} = s_{4,1}x_1^{(k)} + s_{4,2}x_2^{(k)} + s_{4,3}x_3^{(k)} + s_{4,4}x_4^{(k-1)} + s_{4,5}x_5^{(k-1)} + u_4, \\ x_5^{(k)} = s_{5,1}x_1^{(k)} + s_{5,2}x_2^{(k)} + s_{5,3}x_3^{(k)} + s_{5,4}x_4^{(k)} + s_{5,5}x_5^{(k-1)} + u_5. \end{cases} \tag{9}$$

В качестве начального приближения можно взять любой вектор $x^{(0)}$. Вычисления членов последовательности $\{x^{(k)}\}$ прекращаются как только

$$\frac{q}{1-q}||x^{(k)} - x^{(k-1)}|| < \varepsilon,$$

а $x^{(k)}$ объявляется решением системы (1) с точностью ε.

Отметим, что систему (9) можно записать в матричном виде:

$$l.x^{(k)} + d.x^{(k)} + r.x^{(k-1)} = b$$

или

$$x^{(k)} = -(l + d)^{-1}.r.x^{(k-1)} + (l + d)^{-1}.b, \tag{10}$$

что позволяет определить условие сходимости итерационного процесса метода Зейделя в соответствии с приведенной теоремой:

$$|| - (l + d)^{-1}.r|| \leqslant q < 1.$$

Отметим, что на практике обычно применяют итерационную формулу (9).

n.1 Подготовка задания для методов Якоби и Зейделя.

Введем нужные команды. Команду вычисления евклидовой нормы матрицы или вектора:

In[1]:= **nr[t_] := Sqrt[Flatten[t].Flatten[t]]**

необязательную команду округления чисел до n знаков после запятой:

In[2]:= **rn[t_, n_] := N[Round[10^n * t]/10^n]**

команду проверки матрицы на преобладание диагональных элементов:

```
In[3]:= prov[m_] := Block[{n = Length[m]},
          If[MemberQ[Table[2m[[i, i]] > Sum[m[[i, j]], {j, 1, n}], {i, 1, n}], False],
          Print["Условие не выполняется"],
          Print["Матрица с преобладанием диаг. элементов."]]]
```

Пример. Рассмотрим следующую расширенную матрицу системы:

```
In[4]:= m = Table[If[i == j, Random[Integer, {2, 5}]+
                   Random[], Random[]], {i, 1, 5}, {j, 1, 6}];
```

и округлим все элементы матрицы до третьего знака после запятой

In[5]:= **m = rn[m, 3]**

Out[5]=

$$\begin{pmatrix} 3.739 & 0.525 & 0.687 & 0.199 & 0.636 & 0.365 \\ 0.794 & 5.933 & 0.627 & 0.941 & 0.385 & 0.291 \\ 0.677 & 0.031 & 5.723 & 0.912 & 0.307 & 0.513 \\ 0.372 & 0.29 & 0.87 & 2.573 & 0.955 & 0.895 \\ 0.132 & 0.048 & 0.268 & 0.696 & 2.496 & 0.682 \end{pmatrix}$$

Введем число строк матрицы:

In[6]:= **n = Length[m]**

Out[6]= 5

матрицу системы и столбец свободных членов

```
In[7]:= a = a[[1; ; n, 1; ; n - 1]];
        b = m[[All, -1]];
```

введем погрешность решения

In[8]:= **e = 0.0001;**

Проверим матрицу a на преобладание диагональных элементов

In[9]:= **prov[a]**

Out[9]= Матрица с преобладанием диаг. элементов.

Решим теперь систему уравнений с расширенной матрицей m двумя способами, методом Якоби и методом Зейделя. Погрешность полученного решения не должна превышать $e = 0.0001$;

n.2 Метод Якоби

Рассмотрим три матрицы, построенные с помощью элементов матрицы a: левую треугольную матрицу l, правую треугольную матрицу r, диагональную матрицу d, такие, что $a = l + d + r$. Такие матрицы можно построить из элементов матрицы a вручную или воспользоваться следующими командами.

In[10]:= **l = a; Do[If[i <= j, l[[i, j]] = 0], {i, 1, n}, {j, 1, n}]; l**

Out[10]=
$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0.794 & 0 & 0 & 0 & 0 \\ 0.677 & 0.031 & 0 & 0 & 0 \\ 0.372 & 0.29 & 0.87 & 0 & 0 \\ 0.132 & 0.048 & 0.268 & 0.696 & 0 \end{pmatrix}$$

In[11]:= $\mathbf{r = a; Do[If[i >= j, r[[i, j]] = 0], \{i, 1, n\}, \{j, 1, n\}]; r}$

Out[11]=
$$\begin{pmatrix} 0 & 0.525 & 0.687 & 0.199 & 0.636 \\ 0 & 0 & 0.627 & 0.941 & 0.385 \\ 0 & 0 & 0 & 0.912 & 0.307 \\ 0 & 0 & 0 & 0 & 0.955 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

In[12]:= $\mathbf{d = DiagonalMatrix[Table[a[[i, i]], \{i, 1, n\}]]}$

Out[12]=
$$\begin{pmatrix} 3.739 & 0 & 0 & 0 & 0 \\ 0 & 5.933 & 0 & 0 & 0 \\ 0 & 0 & 5.723 & 0 & 0 \\ 0 & 0 & 0 & 2.573 & 0 \\ 0 & 0 & 0 & 0 & 2.496 \end{pmatrix}$$

Таким образом, представление (7) в методе Якоби примет вид

$$x = -Inverse[d].(l + r).x + Inverse[d].b. \tag{11}$$

Отсюда следует итерационная формула (формула (2)) для метода Якоби:

$$x^{(k)} = -Inverse[d].(l + r).x^{(k-1)} + Inverse[d].b. \tag{12}$$

Отметим, что якобиан преобразования (11) равен $-Inverse[d].(l + r)$. Найдем его норму:

In[13]:= $\mathbf{q = nr[-Inverse[d].(l + r)]}$

Out[13]= 0.751085

Так как $q < 1$, то, в силу теоремы 1, получаем ещё одно подтверждение сходимости итерационной последовательности, построенной по формуле (12).

Начинаем строить итерационную последовательность по формуле (12). Вводим начальное приближение.

In[14]:= $\mathbf{x = \{1, 1, 1, 1, 1\}}$; (* $x^{(0)} = x$ *)

и запомним его:

In[15]:= $\mathbf{z = x;}$

Выполним следующую команду, которая даст нам второй элемент итерационной последовательности

In[16]:= $\mathbf{x = -Inverse[d].(l + r).x + Inverse[d].b}$ (* $x^{(1)} = x$ *)

Out[16]= {-0.449853, -0.413956, -0.247073, -0.618733, -0.185096}

Сравним погрешность найденного решения с заданной точностью по формуле (4):

In[17]:= $\mathbf{(q/(1 - q))nr[x - z] < e}$

Out[17]= False

Так как условие $(q/(1 - q))||x^{(1)} - x^{(0)}|| < e$ не выполняется, то нужно повторять все команды, начиная с команды 15, до тех пор, пока команда 17 не вернет True. При этом вектор x будет решением с заданной точностью.

Если применить команду цикла, то метод Якоби будет реализован командам:

In[18]:=
```
x = {1, 1, 1, 1, 1};
Do[z = x;
  x = -Inverse[d].(l + r).x + Inverse[d].b;
  If[nr[x - z] < e,
    Print["Решение x = ", x, " получено на", i, " шаге"];
    Break[]],
 {i, 1, 100}]
```

Out[18]= Решение x={0.0466628, -0.0144319, 0.0323898, 0.259266, 0.19525}
получено на 19 шаге

Вектор невязки решения:

In[19]:=
```
b - a.x
```

Out[19]= {0.00007977,0.0001248,0.00009743,0.00009146,0.00005894}

Невязка решения:

In[20]:=
```
nrx[b - a.x]
```

Out[20]= 0.000208

n.3 Метод Зейделя

Напишем матричное уравнение (11) в координатах. Это можно сделать вручную или автоматизировать процесс следующим образом.

Введем матрицы (8)[2] :

In[14]:=
```
B = -Inverse[d].(l + r)
c = Inverse[d].b
```

Out[14]=
$$\begin{pmatrix} 0. & -0.14 & -0.184 & -0.053 & -0.17 \\ -0.134 & 0. & -0.106 & -0.159 & -0.065 \\ -0.118 & -0.005 & 0. & -0.159 & -0.054 \\ -0.145 & -0.113 & -0.338 & 0. & -0.371 \\ -0.053 & -0.019 & -0.107 & -0.279 & 0. \end{pmatrix}$$

Out[15]= {0.0976197, 0.0490477, 0.0896383, 0.347843, 0.273237}

Следующая команда поможет написать систему уравнений. Напишем матричное равенство (11) в координатах. При этом, округлим коэффициенты для более компактной записи.

In[16]:=
```
Table[x_i == Sum_{j=1}^{n} rn[B[[i, j]], 3] x_j + rn[c[[i]], 3],
 {i, 1, n}]
```

Out[16]=
$$\{x_1 == 0.x_1 - 0.14x_2 - 0.184x_3 - 0.053x_4 - 0.17x_5 + 0.098,$$
$$x_2 == -0.134x_1 + 0.x_2 - 0.106x_3 - 0.159x_4 - 0.065x_5 + 0.049,$$
$$x_3 == -0.118x_1 - 0.005x_2 + 0.x_3 - 0.159x_4 - 0.054x_5 + 0.09,$$
$$x_4 == -0.145x_1 - 0.113x_2 - 0.338x_3 + 0.x_4 - 0.371x_5 + 0.348,$$
$$x_5 == -0.053x_1 - 0.019x_2 - 0.107x_3 - 0.279x_4 + 0.x_5 + 0.273\}$$

[2]Перед выполнением следующей, 14 команды, нужно выполнить 13 первых команд предыдущего пункта.

Перед копированием этой системы, нужно заменить двойные равенства одинарными, в конце каждой строки поставить точку с запятой и убрать фигурные скобки.

Рассмотрим один шаг метода Зейделя. Введем начальное приближение и найдем второй элемент итерационной последовательности, выполнив следующие команды:

```
In[17]:= x1 = 1; x2 = 1; x3 = 1; x4 = 1; x5 = 1;
In[18]:= x1 = 0.x1 − 0.14x2 − 0.184x3 − 0.053x4 − 0.17x5 + 0.098;
         x2 = −0.134x1 + 0.x2 − 0.106x3 − 0.159x4 − 0.065x5 + 0.049;
         x3 = −0.118x1 − 0.005x2 + 0.x3 − 0.159x4 − 0.054x5 + 0.09;
         x4 = −0.145x1 − 0.113x2 − 0.338x3 + 0.x4 − 0.371x5 + 0.348;
         x5 = −0.053x1 − 0.019x2 − 0.107x3 − 0.279x4 + 0.x5 + 0.273;
         {x1, x2, x3, x4, x5, }
Out[18]= {-0.449,-0.220834,-0.0689138,0.0903521,0.283158}
```

Заметим, что при вычислении x_2 было учтено, то, что x_1 уже найдено. Найденные координаты учитываются при вычислении следующих координат. В этом суть метода Зейделя. Повторяя 18 команды, получим следующие элементы итерационной последовательности. Здесь уместно применить команду цикла.

Введем начальное приближение покоординатно и в векторной форме.

```
In[19]:= x1 = 1; x2 = 1; x3 = 1; x4 = 1; x5 = 1;
         xv = {x1, x2, x3, x4, x5};
```

и выполним цикл

```
In[20]:= Do[z = xv;
      x1 = 0.x1 − 0.14x2 − 0.184x3 − 0.053x4 − 0.17x5 + 0.098;
      x2 = −0.134x1 + 0.x2 − 0.106x3 − 0.159x4 − 0.065x5 + 0.049;
      x3 = −0.118x1 − 0.005x2 + 0.x3 − 0.159x4 − 0.054x5 + 0.09;
      x4 = −0.145x1 − 0.113x2 − 0.338x3 + 0.x4 − 0.371x5 + 0.348;
      x5 = −0.053x1 − 0.019x2 − 0.107x3 − 0.279x4 + 0.x5 + 0.273;
      xv = {x1, x2, x3, x4, x5};
      If[nr[xv − z] < e, Print["Решение x = ", xv, " получено на",
       i, " шаге"];
      Break[]], {i, 1, 100}]
Out[20]= Решение x={0.0471535, -0.0147097, 0.0327324, 0.259457, 0.19489}
         получено на 7 шаге
```

Найдем вектор невязки решения:

```
In[21]:= b − a.xv
Out[21]= {-0.00165295, 0.00112842, -0.00224993, -0.00045455, 0.000683309}
```

и невязку решения:

```
In[20]:= nr[b − a.xv]
Out[20]= 0.0031211
```

§9 Градиентные методы решения систем линейных алгебраических уравнений

В данном параграфе рассматриваются два метода, в которых задача нахождения решения системы линейных алгебраических уравнений подменяется экстремальной

задачей, задачей определения точки минимума некоторой вспомогательной квадратичной функции.

n.1 Метод наискорейшего спуска

Данный метод применяется для решения систем вида

$$A.x = b \tag{1}$$

с **положительно определенной симметричной матрицей**. Метод наискорейшего спуска является итерационным методом и заключается в построении последовательности векторов $\{x_k\}$, $k = 0, 1, 2, ...$, где x_0 - любое начальное приближение, сходящейся к вектору ξ - решению системы (1). Для построения такой последовательности рассматривается вспомогательная функция

$$H(x) = (A.x, x) - 2(x, b).$$

Задача нахождения решения системы (1) подменяется экстремальной задачей - задачей определения точки минимума функции $H(x)$. Эта функция является квадратичной функцией относительно координат вектора x. Множества уровня такой функции $H(x) = const$ есть поверхности второго порядка. В двумерном случае - кривые второго порядка.

Лемма. Точка минимума функции $H(x)$ существует и является решением ξ системы уравнений (1).

Доказательство. Так как матрица системы (1) положительно определена, то определитель матрицы системы не равен нуля. Следовательно система имеет единственное решение, которое можно получить, например, с помощью обратной матрицы системы $\xi = A^{-1}.b$.

Пусть $x = \xi + \Delta$ произвольный вектор. Тогда, по свойствам скалярного произведения можно записать, что

$$\begin{aligned}
&H(x) = H(\xi + \Delta) = (A.(\xi + \Delta\ ), \xi + \Delta\ ) - 2(\xi + \Delta, b) = \\
&(A.\xi, \xi) + (A.\xi, \Delta) + (A.\Delta, \xi) + (A.\Delta, \Delta) - 2(\xi, b) - \\
&2(\Delta, b) = H(\xi) + 2(A.\xi, \Delta) - 2(\Delta, b) + (A.\Delta, \Delta) = \\
&H(\xi) + 2(A.\xi - b, \Delta) + (A.\Delta, \Delta) \geqslant H(\xi).
\end{aligned}$$

Здесь учтены равенства: $(A.\xi, \Delta) = (\xi, A.\Delta)$ в силу симметричности матрицы A, $A.\xi - b = 0$ так как ξ есть решение системы (1), а $(A.\Delta, \Delta) > 0$ в силу положительной определенности матрицы A.

Таким образом, $H(\xi) = \min\limits_{x} H(x)$. Лемма доказана.

Найдем градиент ∇H функции $H(x)$. Пусть p - произвольный единичный вектор. Тогда, проводя аналогичные преобразования, получим

$$H(x + tp) = H(x) + 2t(A.x - b, p) + t^2(A.p, p).$$

Отсюда

$$\left.\frac{dH(x + tp)}{dt}\right|_{t=0} = 2(A.x - b, p) = 2|A.x - b||p| \cos\varphi, \tag{2}$$

где φ угол между векторами $A.x - b$ и p. Градиент ∇H есть вектор, в направлении которого функция растет быстрее всего. В направлении градиента производная функции имеет максимальное значение. Из представления производной (2) видно, что производная достигает максимального значения при $\varphi = 0$, то есть тогда, когда вектор p сонаправлен с вектором $A.x - b$. Таким образом, градиент функции

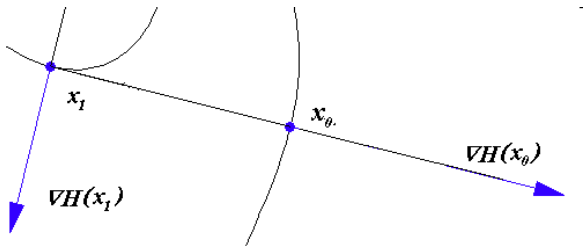
$$\nabla H = A.x - b = -r,$$

то есть градиент функции $H(x)$ есть вектор, противоположный вектору невязки r системы (1) для вектора x.

Элементарный шаг метода наискорейшего спуска выполняется по формуле:

$$x_{k+1} = x_k + \alpha_k r_k, \tag{3}$$

$k = 0, 1, 2, ...$, где x_0 - произвольный вектор - начальное приближение, r_k -вектор, противоположный градиенту функции $H(x)$ в точке x_k, то есть вектор невязки системы (1) в точке x_k.



Функция $H(x)$ убывает при движении в направлении вектора r_k от точки x_k до тех пор, пока луч $x_k + tr_k$, $t > 0$, не коснется первый раз более низкого множества уровня функции $H(x)$. В такой точке касания x_{k+1} функция $H(x)$ достигает минимального значения локально, при этом $H(x_{k+1}) < H(x_k)$. На рисунке показаны два элемента последовательности, x_0 - начальное приближение, произвольный вектор; x_1 - второй элемент последовательности, то есть точка касания лучом $x_0 + tr_0$, $t > 0$, нижележащего множества уровня функции.

Переходя таким образом с одного множества уровня к нижележащему множеству уровня, получим последовательность точек $\{x_k\}$, $k = 0, 1, 2, ...$, такую, что

$$H(x_0) > H(x_1) > ... > H(x_k) > ....$$

Можно доказать (см., например, [7]), что последовательность $\{x_k\}$ сходится к точке минимума функции $H(x)$. В силу леммы, эта точка будет и решением системы (1).

Близость элемента x_k последовательности к решению будем оценивать малостью невязки $r_k = b - A.x_k$ в точке x_k.

Для определения коэффициента α_k в (3) найдем минимум функции $H(x_k + tr_k)$ с аргументом t. Так как

$$H(x_k + tr_k) = H(x_k) + 2t(A.x_k, r_k) + t^2(A.r_k, r_k) - 2t(b, r_k),$$

то, вычислив производную $H(x_k + tr_k)$ по t и приравняв ее к нулю, получим:

$$\alpha_k = \frac{(r_k, r_k)}{(A.r_k, r_k)}. \tag{4}$$

Алгоритм метода наискорейшего спуска:
1). Берем произвольный вектор x_0. Далее для $k = 0, 1, 2, ...$:
2). Находим вектор невязки $r_k = b - A.x_k$. Если невязка $||r_k|| < \varepsilon$, то x_k - ответ.
3). Вычисляем α_k по формуле (4).
4). Находим x_{k+1} по формуле (3) и повторяем выполнение команд, начиная с команды пункта 2). При этом индекс везде увеличиваем на 1.

Рассмотрим пример. Введем матрицу системы

In[1]:= $\mathbf{a} = \{\{\mathbf{1.67}, \mathbf{.32}, \mathbf{.12}, \mathbf{.57}\}, \{\mathbf{.32}, \mathbf{4.17}, \mathbf{.65}, \mathbf{.15}\},$
$\{\mathbf{.12}, \mathbf{.65}, \mathbf{3.15}, \mathbf{.22}\}, \{\mathbf{.57}, \mathbf{.15}, \mathbf{.22}, \mathbf{1.84}\}\};$

столбец свободных членов b и точность, с которой будем искать решение системы $a.x = b$.

In[2]:= $\mathbf{b} = \{\mathbf{1.34}, \mathbf{.85}, \mathbf{1.29}, \mathbf{2.11}\};$
$\varepsilon = \mathbf{0.001};$

Зададим начальное приближение, например, такое:

In[3]:= $\mathbf{x} = \{\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}\};$

Далее, следуя алгоритму, выполняем циклически команды

In[4]:= $\mathbf{r} = \mathbf{b} - \mathbf{a}.\mathbf{x};$
$\alpha = (\mathbf{r}.\mathbf{r})/(\mathbf{a}.\mathbf{r}.\mathbf{r})//\mathbf{N};$
$\mathbf{x} = \mathbf{x} + \alpha\ \mathbf{r}$
$\mathbf{Sqrt}[\mathbf{r}.\mathbf{r}]$

На 13 шаге получим следующее значение x:

Out[17]= {0.432991, 0.0878268, 0.30723, 0.968605}

при невязке равной

0.000656692

что меньше заданной точности $\varepsilon = 0.001$.

Если матрица системы A не симметрична, то части равенства (1) можно домножить на транспонированную матрицу $A^\top$

$$A^\top.A.x = A^\top.b \tag{5}$$

и получить равносильную систему, матрица которой симметрична и положительно определена (теорема Коши-Бине). Система (5) называется нормальной системой.

Задача. Составить команду решающую систему уравнений методом наискорейшего спуска.

n.2 Метод сопряженных градиентов

Данный метод применяется для решения систем вида $A.x = b$ (1) с **положительно определенной симметричной матрицей** и является точным методом. Пусть n-размерность матрицы системы.

Алгоритм метода сопряженных градиентов. Выбираем произвольный вектор x_0 и строим последовательность векторов $\{x_k\}$, невязок $\{r_k = b - A.x_k\}$ и векторов $\{p_k\}$ таких, что

$$p_0 = r_0,$$

а при k=0,1,2,...:

$$\alpha_k = \frac{(r_k, p_k)}{(A.p_k, p_k)},$$

$$\begin{aligned} x_{k+1} &= x_k + \alpha_k p_k, \\ r_{k+1} &= r_k - \alpha_k A.p_k, \\ \beta_k &= -\frac{(A.p_k, r_{k+1})}{(A.p_k, p_k)}, \\ p_{k+1} &= r_{k+1} + \beta_k p_k. \end{aligned} \tag{2}$$

Найдется $k \leqslant n$ такое, что невязка $r_k \approx 0$ (если нет округлений, то $r_k = 0$). Соответствующее x_k будет достаточно точным решением системы.

Покажем как вычислить коэффициенты и поясним суть метода.

Коэффициент β_k определяется из требования сопряженности вектора p_{k+1} с любым вектором p_i, $i \leqslant k$, то есть определяется из равенства нулю скалярных произведений $(A.p_i, p_{k+1}) = 0$, $i \leqslant k$. Коэффициент α_k минимизирует функционал $H(x_k + \alpha p_k)$, где $H(x) = (A.x, x) - 2(x, b)$ (см. предыдущий параграф).

Отметим, что

$$(p_i, A.p_j) = (A.p_i, p_j) = 0 \tag{2}$$

при $i \neq j$, что следует из построения векторов p_i и симметричности матрицы A.

Далее запишем, что

$$(r_i, p_j) = (r_{i-1} - \alpha_{i-1} A.p_{i-1}, p_j) = (r_{i-1}, p_j) - \alpha_{i-1}(A.p_{i-1}, p_j). \tag{3}$$

Пусть $i > j$. Из (3) при $i = j + 1$ получим, что

$$(r_{j+1}, p_j) = (r_j, p_j) - \alpha_j (A.p_j, p_j) = (r_j, p_j) - \frac{(r_j, p_j)}{(A.p_j, p_j)}(A.p_j, p_j) = 0. \tag{4}$$

Если $i > j + 1$, то в силу равенств (2) можно записать, что

$$(r_i, p_j) = (r_{i-1} - \alpha_{i-1} A.p_{i-1}, p_j) = (r_{i-1}, p_j) - \alpha_{i-1}(A.p_{i-1}, p_j) = (r_{i-1}, p_j).$$

Аналогично можно показать, что $(r_{i-1}, p_j) = (r_{i-2}, p_j)$ и так далее. В результате получим

$$(r_i, p_j) = (r_{i-1}, p_j) = ... = (r_{j+1}, p_j) = \ 0$$

в силу (4).

Теперь рассмотрим произведения (r_i, r_j), $i \neq j$. При $i > j$

$$(r_i, r_j) = (r_i, p_j - \beta_{j-1} p_{j-1}) = (r_i, p_j) - \beta_{j-1}(r_i, p_{j-1}) = 0$$

по доказанному. Таким образом, $r_0, r_1, ...$ - ортогональная система векторов. Так как в n-мерном пространстве таких векторов не больше n, то найдется $k \leqslant n$ такое, что $r_k = 0$. Но, $r_k = b - A.x_k = 0$, следовательно x_k - точное решение системы. При этом, k может быть меньше порядка матрицы A. С другой стороны, в силу округлений число шагов, необходимых для достижения заданной точности, может быть больше n. Поэтому данный метод относят и к итерационным методам.

Рассмотрим пример. Введем матрицу с положительными диагональными элементами и с преобладанием диагональных элементов. Такая матрица будет положительно определенной.

In[1]:= $\mathbf{a = Table[If[i == j, Random[Integer, \{5, 10\}] + Random[],}$

Random[]], {**i**, **1**, **5**}, {**j**, **1**, **5**}];

Сделаем ее симметричной:

Table[**a**[[**j**, **i**]] = **a**[[**i**, **j**]], {**i**, **1**, **5**}, {**j**, **i**, **5**}];

Введем столбец свободных элементов:

b = {**1.34**, **.85**, **1.29**, **2.11**, **2.34**};

Решаем систему (1) с заданными коэффициентами методом сопряженных градиентов, то есть реализуем в цикле набор команд (2).

Введем начальное приближение (x_0), начальный вектор ($p_0 = r_0$):

In[2]:= **x** = {**0**, **0**, **0**, **0**, **0**};

p = **r** = **b** − **a**.**x**;

Теперь последовательно выполняем следующие команды:

In[3]:= **q** = **a**.**p**//**N**; (* Повторяющееся произведение вычислим один раз. *)

α = (**r**.**p**)/(**q**.**p**)//**N**;

x = **x** + α **p**

r = **r** − α **q**;

β = −(**q**.**r**)/(**q**.**p**)//**N**;

p = **r** + β **p**;

Sqrt[**r**.**r**] (* Невязка вектора x. *)

После пятой итерации вектор x будет равен

Out[3]= {0.102406,0.0453142,0.080692,0.221428,0.279761}

при невязке равной

0.00028037

Для сравнения выполним встроенную команду решения системы уравнений:

In[4]:= **LinearSolve**[**a**, **b**]//**N**

Out[4]= {0.10238,0.0453067,0.0807048,0.221434,0.279767}

§10 Системы линейных алгебраических уравнений с прямоугольной матрицей

Рассмотрим систему

$$A.x = b \tag{1}$$

с прямоугольной матрицей A порядка $m \times n$, (m - число строк, n число столбцов). Если ранг $\mathrm{rank}(A) = n = m$ и равен рангу расширенной матрицы, то система (1) имеет единственное решение. Решения таких систем рассматривались в предыдущих пунктах. Рассмотрим остальные возможные случаи.

n.1 Обобщенное решение переопределенных систем

Пусть $\mathrm{rank}(A) = n < m$. Система (1) переопределена и может не иметь решения. В этом случае ищется, так называемое, обобщенное решение, то есть такой вектор x, который минимизирует невязку вектора x:

$$r = ||b - A.x||.$$

Оказывается, что обобщенное решение можно найти используя первую трансформацию Гаусса системы (1). Она заключается в умножении матричного уравнения (1) слева на на транспонированную матрицу $A^\top$:

$$A^\top.A.x = A^\top.b \tag{2}$$

Как известно, матрица $A^\top.A$ такой системы является квадратной положительно определенной симметричной матрицей ранга $\mathrm{rank}(A)$. Ее решение и есть обобщенное решение системы (1), его можно получить используя прямые или итерационные методы или, для малых размерностей, с помощью обратной матрицы:

$$x = (A^\top.A)^{-1}.A^\top.b \tag{3}$$

Докажем, что вектор, дающий минимум невязки системы (1), есть решение системы (2). Пусть $A = (a_{ij})$, $i = 1, ..., m$, $j = 1, ..., n$, $b = (b_1, ..., b_m)^\top$, $x = (x_1, ..., x_n)^\top$. Тогда вектор невязки системы (1) равен

$$b - A.x = (b_1 - \sum_{i=1}^{n} a_{1i}x_i, ..., b_m - \sum_{i=1}^{n} a_{mi}x_i).$$

Найдем минимум невязки $||b - A.x||$, как функции от координат вектора x. Если в качестве нормы вектора возьмем евклидову норму, то

$$||b - A.x|| = \sqrt{\sum_{k=1}^{m}(b_k - \sum_{i=1}^{n} a_{ki}x_i)^2}.$$

Достаточно найти минимум функции, стоящей под корнем. Обозначим ее $\varphi(x_1, ..., x_n)$. Как известно из анализа, минимум функции $\varphi(x_1, ..., x_n)$ определяется из системы

$$\frac{\partial\varphi(x_1, ..., x_n)}{\partial x_i} = 0, \quad i = 1, ..., n. \tag{4}$$

Так как

$$\frac{\partial\varphi(x_1, ..., x_n)}{\partial x_i} = 2\sum_{k=1}^{m}(b_k - \sum_{i=1}^{n} a_{ki}x_i)a_{ki}, \quad i = 1, ..., n,$$

то система (4) равносильна системе

$$\sum_{k=1}^{m} a_{ki}b_k = \sum_{k=1}^{m}\sum_{i=1}^{n} a_{ki}a_{kj}x_i, \quad i = 1, ..., n. \tag{5}$$

Теперь легко проверить, что $A^\top.A = (\sum_{k=1}^{m} a_{ki}a_{kj})$, $A^\top.A.x$ совпадает с правой частью (5), а $A^\top.b$ совпадает с левой частью (5). Таким образом, система (2) есть матричная запись системы (5). Но, решение системы (5) есть минимум невязки системы (1). Следовательно вектор, минимизирующий невязку системы (1), есть решение системы (2).

Рассмотрим пример. Введем матрицу порядка (6×3) и найдем ее ранг:

In[1]:= **a = Table[Random[Integer, {4, 10}], {i, 1, 6}, {j, 1, 3}]**

Out[1]=

$$\begin{pmatrix} 6 & 8 & 6 \\ 8 & 7 & 5 \\ 8 & 8 & 8 \\ 4 & 9 & 8 \\ 10 & 4 & 4 \\ 6 & 5 & 6 \end{pmatrix}$$

In[2]:= **MatrixRank[a]**

Out[2]= 3

Введем столбец свободных элементов:

In[3]:= **b = Table[Random[Integer, {4, 10}], {i, 1, 6}]**

Out[3]= {10,9,6,9,6,6}

Рассмотрим расширенную матрицу системы (1):

In[6]:= **m = Transpose[Append[Transpose[a], b]]**

Out[6]=

$$\begin{pmatrix} 6 & 8 & 6 & 10 \\ 8 & 7 & 5 & 9 \\ 8 & 8 & 8 & 6 \\ 4 & 9 & 8 & 9 \\ 10 & 4 & 4 & 6 \\ 6 & 5 & 6 & 6 \end{pmatrix}$$

и ее ранг

In[7]:= **MatrixRank[m]**

Out[7]= 4

Так как ранги матрицы и расширенной матрицы системы (1) не равны, то система не имеет решения.

Составим систему (2). Матрица системы (2):

In[4]:= **Transpose[a].a**

Out[4]=

$$\begin{pmatrix} 316 & 274 & 248 \\ 274 & 299 & 241 \\ 248 & 265 & 241 \end{pmatrix}$$

Столбец свободных членов системы (2):

In[5]:= **Transpose[a].b**

Out[5]= {312,326,285}

Найдем обобщенное решение системы (2) по формуле (3):

In[8]:= **x = Inverse[Transpose[a].a].Transpose[a].b//N//Chop**

Out[8]= {0.273413,1.61131,-0.870558}

Найдем вектор невязки решений:

In[9]:= **b − a.x**

Out[9]= {0.692363, -0.113708, -2.11335, 0.368991, 0.302848, 1.5263}

Как было отмечено, норма этого вектора не больше нормы вектора невязки произвольного вектора.

n.2 Нормальное решение недоопределенных систем

Пусть $\mathrm{rank}(A) = m < n$. Система (1) недоопределенная система. Такая система может иметь бесконечно много решений. С помощью второй трансформации Гаусса можно найти нормальное решение x_0 системы, то есть решение системы, имеющее минимальную (евклидову) норму среди всех решений. Для этого решим вспомогательную систему $A.A^\top.y = b$ (вторая трансформация Гаусса системы (1)), а затем находим решение системы (1): $x = A^\top.y$. Решение первой системы можно получить с помощью обратной матрицы $y = (A.A^\top)^{-1}.b$. Тогда

$$x_0 = A^\top.(A.A^\top)^{-1}.b \tag{4}$$

Действительно, пусть $u_1, ..., u_{n-m}$ - ортонормированное фундаментальное решение системы. Тогда скалярное произведение $(x_0, u_i) = (A^\top.y, u_i) = (y, A.u_i) = 0$ для всех $i = 1, ..., n-m$, так как все векторы u_i образуют базис нуль-пространства матрицы A.

Так как x_0 - частное решение системы, то

$$x = x_0 + \sum_{k=1}^{n-m} c_k u_k$$

- общее решение системы (1). Отсюда получаем

$$||x||^2 = (x, x) = (x_0 + \sum_{k=1}^{n-m} c_k u_k)^2 = x_0^2 + 2\sum_{k=1}^{n-m} c_k (u_k, x_0) +$$

$$\sum_{k,l=1}^{n-m} c_k^2 (u_k, u_l) = x_0^2 + \sum_{k=1}^{n-m} c_k^2$$

Минимум нормы $||x||$ достигается при $c_1 = ... = c_{n-m} = 0$, то есть при $x = x_0$.

Пример прост. Возьмем матрицу и найдем ее ранг.

In[1]:=
```
a = Table[Random[Integer, {4, 10}], {i, 1, 3}, {j, 1, 6}]
MatrixRank[a]
```

Out[1]=

$$\begin{pmatrix} 9 & 6 & 9 & 9 & 8 & 8 \\ 8 & 6 & 9 & 9 & 10 & 9 \\ 10 & 6 & 10 & 6 & 9 & 10 \end{pmatrix}$$

3

Столбец свободных членов.

In[2]:=
```
b = Table[Random[Integer, {4, 10}], {i, 1, 3}]
```

Out[2]= {6,6,10}

Нормальное решение находим по формуле (4):

In[3]:=
```
x = Transpose[a].Inverse[a.Transpose[a]].b//N
```

Out[3]= {0.529096,0.06451,0.367066,-0.714138,0.0322255,0.464585}

n.3 Обобщенно-нормальное решение

Пусть $r = \mathrm{rank}(A) < \min(m, n)$. В этом случае ищется обобщенно-нормальное решение, то есть такой вектор, норма которого наименьшая среди всех векторов, достовляющих минимальную норму вектора невязки. Такое решение можно найти, если известно разложение $A = B.C$, где матрица B порядка $m \times r$, матрица C порядка $r \times n$ и обе матрицы имеют ранг r. В этом случае система (1) принимает вид

$$B.C.x = b \tag{5}$$

Обозначим $C.x = y$ и получим систему равносильную уравнению (5):

$$\begin{cases} B.y = b, \\ C.x = y \end{cases} \tag{6}$$

Первое матричное уравнение системы - переопределенная система: $m > r$, второе уравнение - недоопределенная система: $r < n$. Возьмем в качестве решения первой системы (6) обобщенное решение системы: $y = (B^\top.B)^{-1}.B^\top.b$ и $x = C^\top.(C.C^\top)^{-1}.y$ - нормальное решение второй системы (6). Тогда обобщенно-нормальное решение

$$x = C^\top.(C.C^\top)^{-1}.(B^\top.B)^{-1}.B^\top.b \tag{7}$$

Задача. Ввести систему уравнений и найти обобщенно-нормальное решение.

n.4 Псевдообратные матрицы

Для прямоугольных, в частности, квадратных вырожденных матриц A вводится псевдо-обратная матрица (или обобщенно-обратная матрица) A^+ следующими равенствами:

$$\begin{gathered} A.A^+.A = A, \\ A^+.A.A^+ = A^+, \\ (A.A^+)^\top = A.A^+, \\ (A^+.A)^\top = A^+.A. \end{gathered}$$

Если матрица A обратима, то $A^+ = A^{-1}$.

Решение системы $A.x = b$ с прямоугольной матрицей теперь можно записать в виде $x = A^+.b$

Для доказательства этого утверждения сделаем предположение, что матрица системы имеет вида $A = B.C$, где матрица B порядка $m \times r$, матрица C порядка $r \times n$ и обе матрицы имеют ранг r.

Непосредственной проверкой убеждаемся, что

$$A^+ = C^\top.(C.C^\top)^{-1}.(B^\top.B)^{-1}.B^\top.$$

Пусть $B = A$ а $C = E$ - единичная матрица, тогда $A^+ = (A^\top.A)^{-1}.A^\top$, а $x = A^+.b = (A^\top.A)^{-1}.A^\top.b$ - обобщенное решение.

Пусть $C = A$ а $B = E$ - единичная матрица, тогда $A^+ = A^\top.(A.A^\top)$, а $x = A^+.b = A^\top.(A.A^\top).b$ - нормальное решение.

Третий случай следует из равенства (7).

Задача. Найти решения систем уравнений этого параграфа, применяя псевдо-обратную матрицу.

§11 Обращение матриц

Рассмотрим два метода, позволяющие эффективно находить обратную матрицу для невырожденной матрицы $A = (a_{ij})$ порядка n.

n.1 Метод окаймления

Запишем матрицу A в виде следующих блоков

$$A = \left(\begin{array}{c|c} A_{n-1} & u_n \\ \hline v_n & a_{nn} \end{array} \right),$$

где вектор-столбец $u_n = (a_{1,n}, a_{2,n}, ..., a_{n-1,n})^\top$, вектор-строка $v_n = (a_{n,1}, a_{n,2}, ..., a_{n,n-1})$, а матрица

$$A_{n-1} = \begin{pmatrix} a_{1,1} & a_{1,2} & ... & a_{1,n-1} \\ a_{2,1} & a_{2,2} & ... & a_{2,n-1} \\ \vdots & \vdots & ... & \vdots \\ a_{n-1,1} & a_{n-1,2} & ... & a_{n-1,n-1} \end{pmatrix}.$$

Метод окаймления позволяет найти A^{-1}, если известна обратная матрица A_{n-1}^{-1}. Покажем, как это сделать. Будем искать обратную матрицу в следующем блочном виде:

$$A^{-1} = \left(\begin{array}{c|c} P_{n-1} & r_n \\ \hline q_n & \frac{1}{\alpha_n} \end{array} \right), \tag{1}$$

где P_{n-1} квадратная матрица порядка $n-1$, r_n-вектор-столбец и q_n-вектор-строка размерности $n-1$, α_n - число.

По определению обратной матрицы должно выполняться равенство $A.A^{-1} = E$. Применяя правило умножения блочных матриц, запишем последнее равенство в виде:

$$\left(\begin{array}{c|c} A_{n-1}.P_{n-1} + u_n.q_n & A_{n-1}.r_n + \frac{1}{\alpha_n}u_n \\ \hline v_n.P_{n-1} + a_{nn}q_n & v_n.r_n + a_{nn}\frac{1}{\alpha_n} \end{array} \right) = \left(\begin{array}{c|c} E' & 0 \\ \hline 0 & 1 \end{array} \right).$$

Здесь E' - единичная матрица порядка $n-1$. Приравнивая соответствующие блоки, получаем

$$A_{n-1}.P_{n-1} + u_n.q_n = E' \quad (a) \qquad A_{n-1}.r_n + \frac{1}{\alpha_n}u_n = 0 \quad (c)$$

$$v_n.P_{n-1} + a_{nn}q_n = 0 \qquad (b) \qquad v_n.r_n + a_{nn}\frac{1}{\alpha_n} = 1 \quad (d)$$

Разрешим эти уравнения. Из (c) найдем

$$r_n = -\frac{1}{\alpha_n}A_{n-1}^{-1}.u_n \tag{2}$$

и подставим найденное r_n в равенство (d)

$$v_n.(-\frac{1}{\alpha_n}A_{n-1}^{-1}.u_n) + a_{nn}\frac{1}{\alpha_n} = 1,$$

откуда найдем

$$\alpha_n = a_{nn} - v_n.A_{n-1}^{-1}.u_n. \tag{3}$$

Далее, из равенство (a)

$$P_{n-1} = A_{n-1}^{-1} - A_{n-1}^{-1}.u_n.q_n. \tag{4}$$

Подставляя P_{n-1} в равенство (b), получим

$$v_n.A_{n-1}^{-1} - v_n.A_{n-1}^{-1}.u_n.q_n + a_{nn}q_n = 0$$

или, с учетом (3),

$$v_n.A_{n-1}^{-1} + (\alpha_n - a_{nn}).q_n + a_{nn}q_n = 0.$$

Из этого равенства находим

$$q_n = -\frac{1}{\alpha_n}v_n.A_{n-1}^{-1}.$$

Найденное q_n подставляем в (4) и находим последний блок обратной матрицы

$$P_{n-1} = A_{n-1}^{-1} + \frac{1}{\alpha_n}A_{n-1}^{-1}.u_n.v_n.A_{n-1}^{-1}.$$

Теперь обратная матрица A^{-1} представима в следующем виде:

$$A^{-1} = \left(\begin{array}{c|c} A_{n-1}^{-1} + \frac{1}{\alpha_n}A_{n-1}^{-1}.u_n.v_n.A_{n-1}^{-1} & -\frac{1}{\alpha_n}A_{n-1}^{n-1}.u_n \\ \hline -\frac{1}{\alpha_n}v_n.A_{n-1}^{-1} & \frac{1}{\alpha_n} \end{array} \right). \tag{5}$$

где α_n вычисляется по формуле (3).

Таким образом, если нам известна обратная матрица главного минора $n-1$ порядка A_{n-1}^{-1}, то можем найти и A^{-1}. Для нахождения A_{n-1}^{-1} по формуле (5), нужно знать обратную матрицу A_{n-2}^{-1} главного минора $n-2$ порядка матрицы A. Проводя аналогичные рассуждения, получим, что для определения A_2^{-1} нужно знать A_1^{-1}. Но, $A_1^{-1} = (1/a_{11})$.

Отсюда вытекает следующий алгоритм вычисления обратной матрицы. Сначала находим $A_1^{-1} = (1/a_{11})$. Применяя теперь формулу (5), находим A_2^{-1}. Зная A_2^{-1}, по формуле (5) находим A_3^{-1} и так далее. Через $n-1$ шаг, зная уже A_{n-1}^{-1}, найдем A^{-1} по формуле (5).

Рассмотренная схема нахождения обратной матрицы A^{-1} называется методом "окаймления". Для реализации метода окаймления необходимо, чтобы матрицы всех главных миноров матрицы A были обратимы. Как известно из алгебры, положительно определенные матрицы обладают такими свойствами.

Рассмотрим применение метода окаймления к нахождению обратной матрицы к матрице

In[1]:= $\mathbf{a} = \{\{-\mathbf{3}, \mathbf{4}, \mathbf{5}, \mathbf{4}\}, \{\mathbf{2}, \mathbf{3}, -\mathbf{1}, \mathbf{1}\}, \{\mathbf{3}, \mathbf{5}, -\mathbf{1}, \mathbf{2}\}, \{\mathbf{3}, -\mathbf{1}, \mathbf{4}, \mathbf{1}\}\}$

Можно проверить условие существования обратной матрицы для матрицы m: $Det[m] \neq 0$.

Введем размерность матрицы

In[1]:= $\mathbf{n} = \mathbf{Length}[\mathbf{a}]$;

Обратная матрица к матрицы главного минора порядка 1, то есть к матрице $\{\{-3\}\}$ равна

In[2]:= $\mathbf{t} = \{\{\mathbf{1}/\mathbf{a}[[\mathbf{1},\mathbf{1}]]\}\}$

Out[2]= $\{\{-\frac{1}{3}\}\}$

Пусть

In[3]:= $\mathbf{k} = \mathbf{1}$;

Матрица-столбец, состоящий из k первых элементов $k+1$ столбца:

In[4]:= $\mathbf{u} = \mathbf{a}[[\mathbf{1};;\mathbf{k},\mathbf{k}+\mathbf{1};;\mathbf{k}+\mathbf{1}]]$;

Out[4]= $\{\{4\}\}$

Матрица-строка, состоящая из k первых элементов $k+1$ строки:

In[5]:= $\mathbf{v} = \mathbf{a}[[\mathbf{k}+\mathbf{1};;\mathbf{k}+\mathbf{1},\mathbf{1};;\mathbf{k}]]$;

Out[5]= $\{\{2\}\}$

Обозначим через p и w повторяющиеся части в формуле (1) $A_{n-1}^{-1}.u_n$ и $v_n.A_{n-1}^{-1}$ соответственно:

In[6]:= $\mathbf{p} = \mathbf{t}.\mathbf{u}$;

Out[6]= $\{\{-\frac{4}{3}\}\}$

In[7]:= $\mathbf{w} = \mathbf{v}.\mathbf{t}$;

Out[7]= $\{\{-\frac{2}{3}\}\}$

Вычисляем коэффициент α :

In[8]:= $\alpha = (\mathbf{a}[[\mathbf{k}+\mathbf{1},\mathbf{k}+\mathbf{1}]] - \mathbf{v}.\mathbf{p})[[\mathbf{1},\mathbf{1}]]//\mathbf{N}$;

Out[8]= 5.66667

Первый блок в первой строке (1) (то есть P_{n-1}):

In[9]:= $\mathbf{b} = \mathbf{t} + \frac{\mathbf{1}}{\alpha}\mathbf{p}.\mathbf{w}//\mathbf{N}$;

Out[9]= {{-0.176471}}

столбец r :

In[10]:= $\mathbf{r} = -\frac{\mathbf{1}}{\alpha}\mathbf{p}$;

Out[10]= $\{\{0.235294\}\}$

строка q :

In[11]:= $\mathbf{q} = -\frac{\mathbf{1}}{\alpha}\mathbf{w}$;

Out[11]= $\{\{0.117647\}\}$

Формируем обратную матрицу (A_2^{-1}), для этого окаймляем матрицу t порядка k снизу вектором q, справа столбцом r, добавленным снизу элементом $\frac{1}{\alpha}$. Для этого вводим нулевую квадратную матрицу t размера $k+1$ и заменяем в ней блоки в соответствии с формулой (1)[1]:

In[12]:= $\mathbf{t} = \mathbf{Table}[\mathbf{0},\{\mathbf{i},\mathbf{1},\mathbf{k}+\mathbf{1}\},\{\mathbf{j},\mathbf{1},\mathbf{k}+\mathbf{1}\}]$;
$\mathbf{t}[[\mathbf{1};;\mathbf{k},\mathbf{1};;\mathbf{k}]] = \mathbf{b}$;
$\mathbf{t}[[\mathbf{1};;\mathbf{k},\mathbf{k}+\mathbf{1};;\mathbf{k}+\mathbf{1}]] = \mathbf{r}$;
$\mathbf{t}[[\mathbf{k}+\mathbf{1};;\mathbf{k}+\mathbf{1},\mathbf{1};;\mathbf{k}]] = \mathbf{q}$;
$\mathbf{t}[[\mathbf{k}+\mathbf{1},\mathbf{k}+\mathbf{1}]] = \mathbf{1}/\alpha$;

In[13]:= $\mathbf{t}$

Out[13]=
$$\begin{pmatrix} -0.176471 & 0.235294 \\ 0.117647 & 0.176471 \end{pmatrix}$$

В результате получили обратную матрицу к матрице главного минора порядка $k+1=2$, то есть матрицу A_2^{-1}. Теперь увеличиваем k на единицу

In[14]:= $\mathbf{k} = \mathbf{k}+\mathbf{1}$

[1] В программе Mathematica 5 здесь и далее вместо пяти команд 12 можно применить одну команду $t = Transpose[Append[Transpose[Join[b,q]], Append[r,\{\frac{1}{\alpha}\}]//Flatten]]$;

и выполняем все команды с номерами 4-14. Продолжая выполнять циклически команды 4-14, через $n-2$ шага получим обратную матрицу $t = A^{-1}$.

Объединим повторяющиеся команды в цикл.

Введем матрицу размерности, например, 7:

In[1]:=
```
n = 7;
a = Table[Random[], {i, 1, n}, {j, 1, n}];
t = {{1/a[[1, 1]]}}
Do[
  u = a[[1;; k, k + 1;; k + 1]];
  v = a[[k + 1;; k + 1, 1;; k]];
  p = t.u;
  w = v.t;
  α = (a[[k + 1, k + 1]] - v.p)[[1, 1]]//N;
  b = t + 1/α p.w//N;
  r = -1/α p;
  q = -1/α w;
  t = Table[0, {i, 1, k + 1}, {j, 1, k + 1}];
  t[[1;; k, 1;; k]] = b;
  t[[1;; k, k + 1;; k + 1]] = r;
  t[[k + 1;; k + 1, 1;; k]] = q;
  t[[k + 1, k + 1]] = 1/α,
{k, 1, n - 1}]
```

Выводим ответ:

In[2]:=
```
t//Chop//MatrixForm
```

Out[2]=
$$\begin{pmatrix} 0.75 & -4.25 & 1.83333 & 0.583333 \\ 0 & -1. & 0.333333 & 0.333333 \\ -0.25 & 1.75 & -0.833333 & -0.0833333 \\ -1.25 & 4.75 & -1.83333 & -0.0833333 \end{pmatrix}$$

Можно сделать проверку:

In[3]:=
```
a.t//Chop//MatrixForm
```

Out[3]=
$$\begin{pmatrix} 1. & 0 & 0 & 0 \\ 0 & 1. & 0 & 0 \\ 0 & 0 & 1. & 0 \\ 0 & 0 & 0 & 1. \end{pmatrix}$$

Задача. Составить команду вычисления обратной матрицы методом окаймления.

n.2 Метод пополнения

Допустим, что невырожденная матрицы $A = (a_{ij})$ порядка n допускает представление в виде:

$$A = B + u.v, \tag{1}$$

где B некоторая невырожденная матрица, u есть n-вектор-столбец, а v - n-вектор-строка.

Тогда

$$A^{-1} = B^{-1} - \frac{1}{\gamma} B^{-1}.u.v.B^{-1}, \tag{2}$$

если $\gamma = 1 + v.B^{-1}.u \neq 0$.

Действительно, $(B + u.v).(B^{-1} - \frac{1}{\gamma}B^{-1}.u.v.B^{-1}) = B.B^{-1} - \frac{1}{\gamma}B.B^{-1}.u.v.B^{-1} + u.v.B^{-1} - \frac{1}{\gamma}u.\underbrace{v.B^{-1}.u}_{\gamma-1}.v.B^{-1} = E - \frac{1}{\gamma}.u.v.B^{-1} + u.v.B^{-1} - \frac{1}{\gamma}(\gamma - 1)u.v.B^{-1} = E - \frac{1}{\gamma}.u.v.B^{-1} + u.v.B^{-1} - u.v.B^{-1} + \frac{1}{\gamma}u.v.B^{-1} = E$.

Теперь можно найти элементы матрицы A^{-1} обратной к $A = (a_{ij})$. Для этого проделаем n однотипных шагов. На первом шаге рассмотрим матрицу

$$A_1 = \begin{pmatrix} a_{11} & a_{12} & ... & a_{1n} \\ 0 & 1 & ... & 0 \\ \vdots & \vdots & ... & \vdots \\ 0 & 0 & ... & 1 \end{pmatrix},$$

которая получается из единичной матрицы заменой первой строки на первую строку матрицы A. Представим матрицу A_1 в виде (1). Такое разложение можно получить из вида матрицы A_1:

$$A_1 = \begin{pmatrix} 1 & 0 & ... & 0 \\ 0 & 1 & ... & 0 \\ \vdots & \vdots & ... & \vdots \\ 0 & 0 & ... & 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} .(a_{11} - 1, a_{12}, ..., a_{1n}), \tag{3}$$

то есть,

$$A_1 = E + u_1.v_1,$$

где E - единичная матрица, u_1 - вектор-столбец, а v_1 - вектор-строка в (3). Так как $E^{-1} = E$, то по формуле (2) запишем:

$$A_1^{-1} = E^{-1} - \frac{1}{\gamma} E^{-1}.u_1.v_1.E^{-1}, \tag{4}$$

где $\gamma = 1 + v_1.E^{-1}.u_1 = 1 + v_1.u_1 = 1 + (1.(a_{11} - 1) + 0.a_{12} + ... + 0.a_{n1}) = a_{11}$.

Поэтому

$$A_1^{-1} = E - \frac{1}{a_{11}} u_1.v_1 = \begin{pmatrix} \frac{1}{a_{11}} & -\frac{a_{12}}{a_{11}} & ... & -\frac{a_{1n}}{a_{11}} \\ 0 & 1 & ... & 0 \\ \vdots & \vdots & ... & \vdots \\ 0 & 0 & ... & 1 \end{pmatrix}.$$

На втором шаге пополним матрицу A_1 - заменим в матрице A_1 вторую строку на вторую строку матрицы A:

$$A_2 = \begin{pmatrix} a_{11} & a_{12} & a_{13} & ... & a_{1n} \\ a_{21} & a_{22} & a_{23} & ... & a_{2n} \\ 0 & 0 & 1 & ... & 0 \\ \vdots & \vdots & ... & \vdots & \\ 0 & 0 & 0 & ... & 1 \end{pmatrix}$$

и представим матрицу A_2 в виде (1):

$$A_2 = \begin{pmatrix} a_{11} & a_{12} & ... & a_{1n} \\ 0 & 1 & ... & 0 \\ \vdots & \vdots & ... & \vdots \\ 0 & 0 & ... & 1 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} .(a_{21}, a_{22} - 1, ..., a_{2n}) \qquad (5)$$

или

$$A_2 = A_1 + u_2.v_2,$$

где u_2 - вектор-столбец, а v_2 - вектор-строка в (5). Зная A_1^{-1}, по формуле (2) находим A_2^{-1}. Пополняя матрицу A_2 третьей строкой матрицы A, по аналогии с предыдущими действиями, получим A_3^{-1} и так далее. Исчерпав все строки матрицы A, найдем $A^{-1} = A_n^{-1}$.

Рассмотрим пример. Найдем обратную матрицу к матрице

In[1]:= $\mathbf{a} = \{\{3, -4, 5, 0\}, \{2, -3, 1, 1\}, \{3, -5, -1, 2\}, \{3, -1, 4, 1\}\};$

Размерность матрицы:

In[2]:= $\mathbf{n} = \mathbf{Length}[\mathbf{a}];$

Рассмотрим первый шаг метода пополнения, ориентируясь на формулу (3). Проведем инициализацию двух переменных:

In[3]:= $\mathbf{A} = \mathbf{id} = \mathbf{IdentityMatrix}[\mathbf{n}];$

Зададим номер шага:

In[4]:= $\mathbf{k} = \mathbf{1};$

Через u обозначим вектор-столбец в (3):

In[5]:= $\mathbf{u} = \mathbf{Transpose}[\{\mathbf{id}[[\mathbf{k}]]\}];$

а через v - вектор, k-ую строку матрицы a :

In[6]:= $\mathbf{v} = \{\mathbf{a}[[\mathbf{k}]]\};$

в котором уменьшим на единицу его k-ую координату и получим вектор v в формуле (3):

In[7]:= $\mathbf{v}[[\mathbf{1}, \mathbf{k}]] = \mathbf{v}[[\mathbf{1}, \mathbf{k}]] - \mathbf{1};$

Вычислим константу

In[8]:= $\gamma = (\mathbf{1} + \mathbf{v}.\mathbf{A}.\mathbf{u})[[\mathbf{1}, \mathbf{1}]];$ (* A - единичная матрица при $k = 1$. *)

и найдем обратную матрицу A_1^{-1} по формуле (4). При этом матрицу A_1^{-1} будем обозначать просто A:

In[9]:= $\mathbf{A} = \mathbf{A} - \frac{\mathbf{1}}{\gamma}\mathbf{A}.\mathbf{u}.\mathbf{v}.\mathbf{A}//\mathbf{N}$

В результате получим матрицу A_1^{-1} :

Out[9]=

$$\begin{pmatrix} 0.333333 & 1.33333 & -1.66667 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Увеличиваем k на единицу и выполним команды 5-9 повторно и так далее. Команды 5-9 нужно выполнить n раз, изменяя всякий раз счетчик k. Последний раз в команде 9 слева будет стоять искомая обратная матрица A^{-1}.

Объединяя повторяющиеся команды в цикл, получим:

In[1]:= $\mathbf{n} = \mathbf{Length}[\mathbf{a}];$

```
A = id = IdentityMatrix[n];
Do[u = Transpose[{id[[k]]}];
  v = {a[[k]]};
  v[[1, k]] = v[[1, k]] - 1;
(* Произведение A.u встречается дважды, вычислим его один раз *)
  w = A.u;
  γ = (1 + v.w)[[1, 1]];
  A = A - 1/γ w.v.A//N,
{k, 1, n}]
A//Chop//MatrixForm
```

Out[2]=

$$\begin{pmatrix} 0.75 & -4.25 & 1.84 & 0.58333 \\ 0 & -1 & 0.33333 & 0.33333 \\ -0.25 & 1.75 & -0.833333 & -0.083333 \\ -1.25 & 4.75 & -1.833333 & -0.083333 \end{pmatrix}$$

Можно проверить результат вычислений и перемножить исходную матрицу a и, найденную данным методом, обратную матрицу A^{-1}:

In[32]:=
```
a.A//Chop//MatrixForm
```

В результате должна появиться единичная матрица.

Задача. Составить команду вычисления обратной матрицы методом пополнения.

§12 Проблема собственный значений

Пусть A - вещественная квадратная матрица. Ненулевой вектор x такой, что

$$A.x = \lambda x, \tag{1}$$

при некотором числе λ, называется собственным вектором матрицы A, а число λ (действительное или комплексное) называется собственным значением матрицы A. Найти все пары $\{\lambda, x\}$, удовлетворяющие равенству (1) - это значит решить проблему собственных значений для данной матрицы. Эта задача сводится к решению однородной системы уравнений

$$(A - \lambda E).x = 0.$$

Такая система имеет ненулевое решение, если ее определитель равен нулю

$$\det(A - \lambda E) = 0.$$

Решение последнего уравнения относительно λ связано с определенными трудностями, поэтому такой подход применяется только при малых размерах матрицы.

Под частичной проблемой собственных значений понимается поиск одного или нескольких собственных значений и собственных векторов.

Рассмотрим несколько численных методов решений проблемы собственных значений.

n.1 Частичная проблема собственных значений

Рассмотрим степенной метод для решения частичной проблемы собственных значений. Это итерационный метод, он позволяет определить наибольшее по модулю собственное значение матрицы A и его собственный вектор. Будем считать, что матрица A есть матрица простой структуры, то есть имеет n линейно независимых собственных векторов $U_1, U_2, ..., U_n$, n-порядок матрицы A. Пусть $\lambda_1, ..., \lambda_n$ - соответствующие собственные значения:

$$A.U_i = \lambda_i U_i$$

для всех $i = 1, 2, ..., n$. Пусть Y_0 - произвольный вектор. Построим итерационную последовательность

$$Y_k = A.Y, \quad k = 1, 2, ... \tag{1}$$

Если

$$Y_0 = a_1 U_1 + ... + a_n U_n,$$

то

$$Y_1 = A.Y_0 = a_1 A.U_1 + ... + a_n A.U_n = a_1 U_1 \lambda_1 + ... + a_n U_n \lambda_n,$$

$$Y_2 = A.Y_1 = A^2.Y_0 = a_1 U_1 \lambda_1^2 + ... + a_n U_n \lambda_n^2$$

и так далее,

$$Y_k = a_1 U_1 \lambda_1^k + ... + a_n U_n \lambda_n^k, \tag{2}$$

Пусть $Y_k = (y_{1k}, ..., y_{nk})^\top$, $U_k = (u_{1k}, ..., u_{nk})^\top$. Тогда (2) можно записать в координатах

$$y_{i,k} = a_1 u_{i,1} \lambda_1^k + ... + a_n u_{i,n} \lambda_n^k, \tag{3}$$

i=1,2,...,n - номер координаты вектора. Введем обозначение $c_{i,k} = a_k u_{i,k}$ и перепишем (3) в виде

$$y_{i,k} = c_{i,1} \lambda_1^k + ... + c_{i,n} \lambda_n^k \tag{4}$$

Рассмотрим частные случаи.

1). Будем считать, что наибольшее по модулю собственное число вещественное и

$$|\lambda_1| > |\lambda_2| \geqslant ... \geqslant |\lambda_n| \tag{5}$$

Из (4):

$$\frac{y_{i,k+1}}{y_{i,k}} = \frac{c_{i,1}\lambda_1^{k+1} + ... + c_{i,n}\lambda_n^{k+1}}{c_{i,1}\lambda_1^k + ... + c_{i,n}\lambda_n^k} = \lambda_1 \frac{1 + \frac{c_{i,2}}{c_{i,1}}(\frac{\lambda_2}{\lambda_1})^{k+1} + ... + \frac{c_{i,n}}{c_{i,1}}(\frac{\lambda_n}{\lambda_1})^{k+1}}{1 + \frac{c_{i,2}}{c_{i,1}}(\frac{\lambda_2}{\lambda_1})^k + ... + \frac{c_{i,n}}{c_{i,1}}(\frac{\lambda_n}{\lambda_1})^k}. \tag{6}$$

В силу (5)

$$(\frac{\lambda_i}{\lambda_1})^k \to 0 \quad \text{при} \quad k \to \infty$$

и всех $i = 2, 3, ..., n$, следовательно, при достаточно больших k, из (6)

$$\lambda_1 = \frac{y_{i,k+1}}{y_{i,k}}, \tag{7}$$

а из (2)

$$Y_k = \lambda_1^k(a_1U_1 + a_2U_2(\frac{\lambda_2}{\lambda_1})^k + ... + a_nU_n(\frac{\lambda_n}{\lambda_1})^k)$$

и, следовательно, при достаточно больших k, вектор

$$Y_k = \lambda_1^k a_1 U_1 \tag{8}$$

то есть векторы Y_k и U_1 параллельны.

Найдем максимальное по модулю собственное число и соответствующий собственный вектор матрицы

In[1]:= $\mathbf{a = Table[Random[Integer, \{-20, 40\}], \{i, 1, 5\}, \{j, 1, 5\}]}$

Out[1]=

$$\begin{pmatrix} 30 & -3 & 37 & -3 & 27 \\ -12 & 33 & 4 & 1 & -12 \\ 21 & 4 & 2 & 28 & -12 \\ 37 & 30 & 11 & 31 & 37 \\ 26 & 3 & 15 & -5 & 11 \end{pmatrix}$$

Введем начальное приближение - произвольный вектор:

In[2]:= $\mathbf{y = \{1, 3, -2, 2, 3\};}$

Запоминаем его

In[3]:= $\mathbf{z = y;}$

Out[3]= {1, 3, -2, 2, 3}

Вычисляем следующий элемент итерации

In[4]:= $\mathbf{y = a.y//N;}$

Out[4]= {22., 45., 49., 278., 28.}

Находим все отношения (7). Для этого напишем отношение векторов

In[5]:= $\mathbf{y/z;}$

Out[5]= {22., 15., -24.5, 139., 9.33333}

Повторяем команды 3-5 пока в выводе команды 5 в координатах после запятой знаки не установятся, так что все координаты этого вектора будут иметь несколько одинаковых знаков после запятой. Любая из этих координат будет приближением максимального по модулю собственнного вектора матрицы a. В силу (8) соответствующим собственным вектором будет вектор y. Вектор y можно пронормировать: y/Max[Abs[y]] - поделить все его координаты на его максимальную по модулю координату.

Для исключения деления на ноль в команде 5 заменим команду 5 командой

$\mathbf{tu = Table[If[z[[i]] \neq 0, y[[i]]/z[[i]]], \{i, 1, 5\}]}$

В списке tu координаты, в которых происходит деление на ноль заменяются словом Null.

Объединим команды 3-5 в цикл.

```
In[1]:= y = {1, 3, -2, 2, 3};
        Do[z = y;
        y = a.y//N;
        tu = Table[If[z[[i]] ≠ 0, y[[i]]/z[[i]]], {i, 1, 5}], {30}]
        tu
        y/Max[Abs[y]]
Out[1]= {69.3317, 69.3318, 69.3316, 69.3316, 69.3317}
```

{0.722497, -0.270482, 0.559827, 1., 0.366368}

Цикл проработал 30 раз. Максимальное по модулю собственное число - 69.331. Второй вектор - соответствующий собственный вектор.

Если координаты первого вектора Out[1] сильно колеблются и различны, то возможно, что имеет место следующий случай.

2). Два наибольшие по модулю собственные числа комплексно-сопряженные: $\lambda_1 = re^{i\varphi}$, $\lambda_2 = re^{-i\varphi}$ и

$$|\lambda_1| = |\lambda_2| > |\lambda_3| \geqslant ... \geqslant |\lambda_n|. \tag{7}$$

Так как A в (1) вещественная матрица, начальное приближение вещественный вектор, то все координаты векторов (4) вещественные. Следовательно в (4) числа $c_{i,1}$, $c_{i,2}$ комплексные: $c_{i,1} = Re^{i\alpha}$, $c_{i,2} = Re^{-i\alpha}$. Отсюда и из (4) получим

$$y_{i,k} = 2Rr^k \cos(k\varphi + \alpha) + c_{i,3}\lambda_3^k + ... + c_{i,n}\lambda_n^k.$$

Следовательно с ростом k координата $y_{i,k}$ может сильно меняться по величине и знаку.

Пусть $p = -(\lambda_1 + \lambda_2)$, $q = \lambda_1\lambda_2$. Значит λ_1, λ_2 по теореме Виета есть корни уравнения $t^2 + pt + q = 0$. Найдем p и q. Считаем, что k настолько велико, что $y_{i,k} = c_{i,1}\lambda_1^k + c_{i,2}\lambda_2^k$. Тогда

$$\begin{aligned} &y_{i,k+1} + py_{i,k} + qy_{i,k-1} = c_{i,1}\lambda_1^{k+1} + c_{i,2}\lambda_2^{k+1} + p(c_{i,1}\lambda_1^k + c_{i,2}\lambda_2^k) + \\ &q(c_{i,1}\lambda_1^{k-1} + c_{i,2}\lambda_2^{k-1}) = c_{i,1}\lambda_1^{k-1}(\lambda_1^2 + p\lambda_1 + q) + \\ &c_{i,2}\lambda_2^{k-1}(\lambda_2^2 + p\lambda_2 + q) = 0. \end{aligned}$$

Возьмем два таких уравнения, например, при i=1,2:

$$y_{1,k+1} + py_{1,k} + qy_{1,k-1} = 0, \qquad y_{2,k+1} + py_{2,k} + qy_{2,k-1} = 0. \tag{8}$$

Решаем их и находим коэффициенты p, q, затем решаем квадратное уравнение $t^2 + pt + q = 0$ и находим максимальные по модулю собственные значения.

Для определения соответствующих собственных векторов U_1 и U_2 заметим, что при достаточно больших k:

$$Y_{k+1} - \lambda_2 Y_k = a_1\lambda_1^k(\lambda_1 - \lambda_2)U_1, \quad Y_{k+1} - \lambda_1 Y_k = a_2\lambda_2^k(\lambda_2 - \lambda_1)U_2. \tag{9}$$

Рассмотрим пример. Сгенерируем матрицу

In[1]:= **a = Table[Random[Integer, {−6, 9}], {i, 1, 5}, {j, 1, 5}]**

Out[1]=

$$\begin{pmatrix} 2 & 1 & 4 & 2 & 5 \\ 3 & -5 & -5 & -4 & 0 \\ 5 & 5 & -2 & 9 & 3 \\ -1 & 6 & 0 & -2 & 4 \\ -4 & 3 & 4 & 2 & -3 \end{pmatrix}$$

и проверим наличие у матрицы комплексных максимальных по модулю собственных значений, используя встроенную функцию Eigensystem[a]:

In[2]:= **Eigensystem[a]//N//TableForm**

Out[2]=

$$\begin{array}{cc} -4.74851 + 3.04078\,i & -4.74851 - 3.04078\,i \\ -0.352922 - 0.357852\,i & -0.352922 + 0.357852\,i \\ -0.815025 + 0.133778\,i & -0.815025 - 0.133778\,i \\ -0.327748 + 0.348332\,i & -0.327748 - 0.348332\,i \\ 0.297933 - 0.0926222\,i & 0.297933 + 0.0926222\,i \\ 1 & 1 \end{array}$$

Выведены только два столбца таблицы. В перовой строке первые два собственных значения, комплексно-сопряженные. Столбцы под ними - соответствующие им собственные векторы. Если у матриц нет комплексных собственных значений, то придется генерировать матрицу повторно.

Вводим начальное приближение - любой вектор:

In[3]:= $\mathbf{q = \{2, 2, 3, 4, 5\};}$

Следующий цикл проработает 80 раз. При каждом проходе цикла, начиная со второго, значениями s, z и q будут являться три последовательных вектора итерации.

In[4]:=
```
Do[s = z;
   z = q;
   q = a.q//N,
  {80}]
```

Выведем вектор

In[5]:= $\mathbf{q}$

Out[5]= $\{7.86153 \times 10^{30}, 8.97203 \times 10^{30}, 7.80515 \times 10^{29}, -2.89674 \times 10^{30}, -1.25619 \times 10^{31}\}$

Координаты следующего вектора будут левыми частями уравнений вида (8)

In[6]:= $\mathbf{er = q + x\ z + y\ s}$

Out[6]= $\{-1.05129 \times 10^{30} x + 6.745647 \times 10^{28} y + 7.8615 \times 10^{30},$
$-2.243030 \times 10^{30} x + 3.84841 \times 10^{29} y + 8.972036 \times 10^{30},$
$-8.27124 \times 10^{29} x + 2.2348481 \times 10^{29} y + 7.80515 \times 10^{29},$
$8.06216 \times 10^{29} x - 1.49058 \times 10^{29} y - 2.896748 \times 10^{30},$
$2.78064 \times 10^{30} x - 4.33373 \times 10^{29} y - 1.25619 \times 10^{31}\}$

Возьмем два уравнения вида (8) и решим систему, состоящую из этих уравнений

In[7]:= $\mathbf{w = Solve[\{er[[1]] == 0, er[[2]] == 0\}, \{x, y\}][[1]]}$

Out[7]= $\{x \to 9.55575, y \to 32.3816\}$

Подставим найденные решения x,y в квадратное уравнение $t^2 + x\ t + y = 0$, решим это уравнение относительно t и получим первые два собственных значения максимальных по модулю.

In[8]:= $\mathbf{sol = Solve[(t^2 + x\ t + y)/.w) == 0, t]}$

Out[8]= $\{\{t \to -4.74874 - 3.04064\,i\}, \{\{t \to -4.74874 + 3.04064\,i\}\}$

Для определения соответствующих собственных векторов построим векторы в левой части равенств (9), то есть, выполним команды

In[9]:=
```
(q - sol[[1, 1, 2]] z)/Last[q - sol[[1, 1, 2]] z]//Chop
(q - sol[[2, 1, 2]] z)/Last[q - sol[[2, 1, 2]] z]//Chop
```

В результате получим два вектора с комплексно-сопряженными координатами. Приведем один из этих векторов.

Out[9]= {-0.352904+ 0.357826 i, -0.815024- 0.133765 i,
-0.327772- 0.348312 i, 0.297921+0.092618 i, 1}

Можно сравнить выводы 8 и 9 с выводом 2, второй столбец.

n.2 Метод Гивенса

Метод Гивенса относится к прямым методам. Он приводит любую симметричную (несимметричную) матрицу к подобной ей трехдиагональной (почти треугольной) матрице и основан на подобных преобразованиях исходной матрицы с помощью матриц вращения. Напомним, что матрица $A = (a_{i,j})$, $i, j = 1, 2, ..., n$, называется трехдиагональной, если в каждой строке все элементы отличные от $a_{i,i-1}$, $a_{i,i}$, $a_{i,i+1}$ - нулевые и почти треугольной, если все элементы, отличные от $a_{i,j}$, $i > j - 2$ - нулевые. Например, при $n = 5$, трехдиагональная и почти треугольная матрицы имеют соответственно следующий вид:

$$\begin{pmatrix} * & * & 0 & 0 & 0 \\ * & * & * & 0 & 0 \\ 0 & * & * & * & 0 \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{pmatrix}, \qquad \begin{pmatrix} * & * & 0 & 0 & 0 \\ * & * & * & 0 & 0 \\ * & * & * & * & 0 \\ * & * & * & * & * \\ * & * & * & * & * \end{pmatrix}.$$

Для приведенной таким образом симметричной матрицы к трехдиагональной, методом бисекции можно решить полную или частичную проблему собственных значений, что и будет сделано в следующих пунктах.. Так как исходная симметричная матрица и трехдиагональная подобны, то эти матрицы имеют одинаковый набор собственных значений (см. последний пункт этого параграфа). Следовательно, для исходной матрицы будет также решена проблема собственных значений.

Рассмотрим квадратную матрицу $A = (a_{i,j})$ порядка n. Пусть T_{ij} матрица вращения порядка n (см. §6) с элементами c и s такими, что $c^2 + s^2 = 1$.

Построим последовательность из подобных матриц, с начальной матрицей A, такую, что у последней матрицы в позициях $(1, 3)$, $(1, 4), ..., (1, n), (2, 4), (2, 5)$, ..., $(2, n)$,..., $(n - 2, n)$ стояли бы нули. Другими словами, все элементы последней матрицы в позициях (i, j) таких, что $i+1 < j$ были бы нули, а последняя матрица была бы либо трехдиагональной (матрица A симметричная), либо почти треугольной. Число матриц последовательности равно числу обнуляемых элементов: $(n^2 - 3n + 2)/2$. Последнюю матрицу такой последовательности условно можно записать так:

$$S = (T^{\top}_{n-1,n}...T^{\top}_{2,4}.T^{\top}_{2,3}).A.(T_{2,3}.T_{2,4}...T_{n-1,n}). \tag{1}$$

В этой записи присутствуют все матрицы набора.

Покажем, как вычислить эти матрицы и выбрать элементы матрицы вращения с и s на каждом шаге. Рассмотрим первое подобное преобразование и найдем вторую матрице последовательности:

$$G = T^{\top}_{i,j}.A.T_{i,j} \tag{2}$$

при $i = 2$, $j = 3$. Пусть $B = A.T_{i,j}$, тогда $G = T^{\top}_{i,j}.B$. Все столбцы матрицы A совпадают со столбцами матрицы $B = (b_{ij})$ за исключением i, j столбцов, которые получаются из соответствующих столбцов матрицы A по формулам

$$b_{mi} = ca_{mi} - sa_{mj}, \quad b_{mj} = sa_{mi} + ca_{mj}, \quad m = 1, ..., n, \tag{3}$$

Аналогично, все строки $G = (g_{ij})$ совпадают со строками B, за исключением i, j строк, которые изменяются по формулам:

$$g_{im} = cb_{im} + sb_{jm}, \quad g_{jm} = -sb_{im} + cb_{jm}, \quad m = 1, ..., n. \tag{4}$$

Отметим, что элементы матриц G и B в позиции $(i-1,j)$ совпадают $g_{i-1,j} = b_{i-1,j}$.

Потребуем, чтобы подобное преобразование (2) обнулило элемент $g_{i-1,j}$ матрицы G в позиции $(i-1,j)$. Так как элемент $g_{i-1,j}$ лежит только на одном изменяющимся j-ом столбце, то, в силу (3) и совпадения элементов $g_{i-1,j} = b_{i-1,j}$, получим

$$g_{i-1,j} = b_{i-1,j} = s\, a_{i-1,i} + c\, a_{i-1,j}.$$

Требуем, чтобы

$$g_{i-1,j} = s\, a_{i-1,i} + c\, a_{i-1,j} = 0.$$

Отсюда, с учетом $c^2 + s^2 = 1$, находим

$$c = \frac{a_{i-1,i}}{\sqrt{a_{i-1,i}^2 + a_{i-1,j}^2}}, \qquad s = -\frac{a_{i-1,j}}{\sqrt{a_{i-1,i}^2 + a_{i-1,j}^2}}. \tag{5}$$

Подставим найденные элементы c и s в матрицу вращения в (2) и получим вторую матрицу последовательности G.

Аналогично проводим следующее умножение в (1) и получаем третью матрицу

$$G_1 = T_{i,j}^{\top}.G.T_{i,j} \tag{6}$$

при $i = 2,\ j = 4$. Отметим, что отсюда и из (2), подставляя соответствующие значения индексов, получим часть формулы (1)

$$G_1 = (T_{2,4}^{\top}.T_{2,3}^{\top}).A.T_{2,3}.T_{2,4}$$

Важно отметить, что элементы c и s для матрицы вращения в (6) вычисляются по формулам (5), но только с помощью элементов второй матрицы G.

Продолжая подобным образом, получим последнюю матрицу S, обладающую нужными свойствами.

Рассмотрим пример. Пусть

In[1]:= $\mathbf{n = 5;}$

Введем матрицу

In[2]:= $\mathbf{a = Table[Random[], \{i, 1, n\}, \{j, 1, n\}];}$

и сделаем ее симметричной

In[3]:= $\mathbf{Table[a[[j, i]] = a[[i, j]], \{i, 1, n\}, \{j, i, n\}];}$

Следующий цикл обнуляет элементы матрицы A в позициях (1,3), (1,4),..., (1,n), (2,4),..., (2,n),..., (n-2,n).

In[4]:=
$$\begin{aligned}
&\mathbf{Do[}\\
&\quad \mathbf{t = IdentityMatrix[n];}\\
&\quad \mathbf{c = \frac{a[[i-1, i]]}{Sqrt[a[[i-1, i]]^2 + a[[i-1, j]]^2]}//N;}\\
&\quad \mathbf{s = -\frac{a[[i-1, j]]}{Sqrt[a[[i-1, i]]^2 + a[[i-1, j]]^2]}//N;}\\
&\quad \mathbf{t[[i, i]] = t[[j, j]] = c;}\\
&\quad \mathbf{t[[i, j]] = s;}\\
&\quad \mathbf{t[[j, i]] = -s;}\\
&\quad \mathbf{a = Transpose[t].a.t//Chop//N,}\\
&\mathbf{\{i, 2, n-1\}, \{j, i+1, n\}]}
\end{aligned}$$

Так как матрица **a** симметричная, то в результате получается трехдиагональная матрица.

In[5]:= **a//MatrixForm**

Out[5]=

$$\begin{pmatrix} 6.18749 & 1.13936 & 0 & 0 & 0 \\ 1.13936 & 7.7463 & 1.6942 & 0 & 0 \\ 0 & 1.6942 & 7.81729 & 1.072777 & 0 \\ 0 & 0 & 1.072777 & 7.1669 & -0.150468 \\ 0 & 0 & 0 & -0.150468 & 8.08056 \end{pmatrix}$$

n.3 Метод бисекций решения полной проблемы собственных значений симметричной матрицы

Найдем собственные числа симметричной трехдиагональной матрицы

$$S_n = \begin{pmatrix} \alpha_1 & \beta_2 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \beta_2 & \alpha_2 & \beta_3 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \beta_3 & \alpha_3 & \beta_4 & \cdots & 0 & 0 & 0 \\ . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & \cdots & \beta_{n-1} & \alpha_{n-1} & \beta_n \\ 0 & 0 & 0 & 0 & \cdots & 0 & \beta_n & \alpha_n \end{pmatrix}.$$

Пусть S_k матрица k-го главного минора матрицы S_n, $P_k(\lambda)$ - характеристический многочлен матрицы S_k :

$$P_k(\lambda) = \begin{vmatrix} \alpha_1 - \lambda & \beta_2 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \beta_2 & \alpha_2 - \lambda & \beta_3 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \beta_3 & \alpha_3 - \lambda & \beta_4 & \cdots & 0 & 0 & 0 \\ . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & \cdots & \beta_{k-1} & \alpha_{k-1} - \lambda & \beta_k \\ 0 & 0 & 0 & 0 & \cdots & 0 & \beta_k & \alpha_k - \lambda \end{vmatrix}.$$

Разложим определитель $P_k(\lambda)$ по последней строке, получим следующее рекуррентное соотношение

$$P_k(\lambda) = (\alpha_k - \lambda)P_{k-1}(\lambda) - \beta_k^2 P_{k-2}(\lambda), \quad k = 2, 3, ..., n.$$

Добавим к этому списку еще два многочлена $P_0(\lambda) = 1$ и $P_1(\lambda) = \alpha_1 - \lambda$.

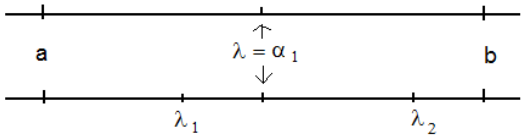
Если $\beta_i \neq 0, i = 2, 3, ..., n$, то корни $\lambda'_1, ..., \lambda'_{k-1}$, полинома $P_{k-1}(\lambda)$ строго разделяют корни $\lambda''_1, ..., \lambda''_k$ полинома $P_k(\lambda)$ для любого $k = 2, 3, ..., n$:

$$\lambda''_1 < \lambda'_1 < \lambda''_2 < \lambda'_2 < \lambda''_3 < ... < \lambda'_{k-1} < \lambda''_k.$$

Все собственные числа матриц S_k, $k \leqslant n$, находятся в пределах от $a = -||S_n||$ и до $b = ||S_n||$ так как из соотношений $S_k.u = \lambda u$, $||u|| = 1$ и очевидного неравенства $||S_k|| \leqslant ||S_n||$ вытекает $|\lambda| = |\lambda|||u|| = ||\lambda\ u|| = ||S_k.u|| \leqslant ||S_k||||u|| \leqslant ||S_k|| < ||S_n||$. Поэтому $-||S_n|| \leqslant \lambda \leqslant ||S_n||$.

Свойство разделения корней многочленов $P_k(\lambda)$ позволяет определить интервалы изоляции корней каждого из многочленов (см. §13, п.1). Так, многочлен $P_1(\lambda) = \alpha_1 - \lambda$ обладает одним корнем $\lambda = \alpha_1$ на отрезке [a,b]. Так как этот корень разделяет корни

λ_1 и λ_2 многочлена $P_2(\lambda)$: $\lambda_1 < \alpha_1 < \lambda_2$, то интервалы изоляции корней многочлена $P_2(\lambda)$ будут



отрезки $[a, \alpha_1]$ и $[\alpha_1, b]$. Найдя корни многочлена $P_2(\lambda)$ находим интервалы изоляции корней многочлена $P_3(\lambda)$ и так далее. Зная интервалы изоляции корней, можно найти корни всех многочленов, например, методом деления отрезка пополам (метод бисекций).

Если $\beta_m = 0$ при некотором $m \leqslant n$, то определитель $P_n(\lambda)$ можно представить как произведение двух определителей и для каждого из них применить метод бисекции.

Найдем собственные числа трехдиагональной симметричной матрицы a из предыдущего параграфа. Введем четыре функции: nr[a] - евклидова норма вектора или матрицы, mg[k] - матрица главного минора матрицы a порядка k, p[k] - характеристический многочлен матрицы mg[k] ($P_k(\lambda)$) и $bis[f, \{a1, b1\}, \varepsilon]$ - команда нахождения корня функции f, обязательно с аргументом λ, методом деления отрезка пополам (§13, п.4), при этом, $\{a1, b1\}$ - отрезок, а ε - точность определения корня.

In[1]:=
```
nr[a_] := Sqrt[(a//Flatten).(a//Flatten)]//N
mg[k_] := a[[1;;k, 1;;k]]
p[k_] := Det[mg[k] - λ IdentityMatrix[k]]
bis[f_, {a1_, b1_}, ε_] :=
Block[{a = a1, b = b1},
   While[Abs[b - a] > ε,
         c = (a + b)/2.;
         If[(f/.λ → a)(f/.λ → c) < 0, b = c,
         If[(f/.λ → c) == 0, Break[], a = c]]
         ];
      c]
```

Найдем интервал $(-||S_n||, ||S_n||)$, содержащий собственные значения матрицы a:

In[2]:=
```
a1 = -nr[a]
b1 = nr[a]
```
Out[2]=
```
-16.9335
 16.9335
```

Зададим точность

In[4]:=
```
ε = 0.0001
```

и найдем корень многочлена $P_1(\lambda)$:

In[5]:=
```
q = bis[p[1], {a1, b1}, ε]
```
Out[5]=
```
6.18748
```

Составим список ss концов интервалов изоляции корней многочлена $P_2(\lambda)$.

In[6]:=
```
ss = ss0 = {a1, q, b1}
```
Out[6]=
```
{-16.9335,6.18748,16.9335}
```

и сформируем список самих интервалов изоляции корней многочлена $P_2(\lambda)$:

In[7]:= **intervals = Partition[ss, 2, 1]**

Out[7]= { {-16.93351, 6.18748}, {6.18748, 16.93351}}

Подействуем функцией bis на каждый такой интервал, при этом, первым аргументом функции будет многочлен $p[2] = P_2(\lambda)$. Напомним, что команда /@ подставляет каждый элемент списка intervals вместо второго аргумента # в команде bis и команда bis находит корень многочлена $p[2]$ на таком интервале. В результате будет составлен список e из корней многочлена $P_2(\lambda)$. Подстановка вместо # список-элемент списка intervals объясняет фигурные скобки в определении команды bis.

In[8]:= **e = bis[p[2], #, ε]&/@intervals**

Out[8]= {5.5864,8.3474}

Составим список ss концов интервалов изоляции корней многочлена $P_3(\lambda)$. Для этого заменим в списке $ss0$ элемент q на полученный список e. Лишние внутренние скобки убираем командой Flatten.

In[9]:= **ss = ss0/.q → e//Flatten**

Out[9]= {-16.9335,5.5864,8.3474,16.9335}

и сформируем список самих интервалов изоляции корней многочлена $P_3(\lambda)$:

In[10]:= **intervals = Partition[ss, 2, 1]**

Out[10]= {{-16.93351, 5.58640},{ 5.58640, 8.34739},{ 8.347395, 16.93351}}

Составим список e из корней многочлена $P_3(\lambda)$:

In[11]:= **e = bis[p[3], #, ε]&/@intervals**

Out[11]= {5.24784,6.83419,9.66913}

Повторяем команды для нахождения корней многочлена $P_4(\lambda)$. Составляем список концов интервалов изоляции корней

In[12]:= **ss = ss0/.q → e//Flatten**

Out[12]= {-16.9335,5.24784,6.83419,9.66913,16.9335}

список самих интервалов изоляции корней многочлена $P_4(\lambda)$:

In[13]:= **intervals = Partition[ss, 2, 1]**

Out[13]= {{ -16.933517, 5.24784},{ 5.24784 6.834189},
{ 6.834189 9.66912},{ 9.66912 16.93351}}

Находим корни многочлена $p[4] = P_4(\lambda)$:

In[14]:= **e = bis[p[4], #, ε]&/@intervals**

Out[14]= {5.14216,6.29266,7.6162,9.86704}

Составляем список концов интервалов изоляции корней $p[5] = P_5(\lambda)$:

In[15]:= **ss = ss0/.q → e//Flatten**

Out[15]= {-16.9335,5.14216,6.29266,7.6162,9.86704,16.9335}

сами интервалы изоляции корней

In[16]:= **intervals = Partition[ss, 2, 1]**

Out[16]= {{-16.93351, 5.14215},{ 5.142157, 6.292658}, { 6.292658 7.616202},
{ 7.61620 9.86704}, { 9.867043 16.93351}}

Находим корни многочлена $p[5] = P_5(\lambda)$, то есть собственные числа трехдиагональной матрицы a :

In[17]:= **e = bis[p[5], #, ε]&/@intervals**

Out[17]= {5.14157,6.28823,7.59253,8.10823,9.86796}

Применим команду цикла для сокращения программы. Перед выполнением этих команд нужно ввести функции nr[a], mg[k], p[k], mg[k] и $bis[f, \{a1, b1\}, \varepsilon]$, матрицу a, погрешность.

```
In[1]:= a1 = −nr[a]; b1 = nr[a];
In[2]:= intervals = {{a1, b1}};
In[3]:= ss0 = {a1, q, b1};
In[4]:= Do[
          e = bis[p[i], #, ε]&/@intervals;
          ss = ss0/.q → e//Flatten;
          intervals = Partition[ss, 2, 1],
        {i, 1, 5}]
In[5]:= e
Out[5]= {5.14164,6.28826,7.59248,8.10819,9.86797}
```

n.4 Метод бисекций решения частичной проблемы собственных значений симметричной матрицы

Последовательность многочленов предыдущего параграфа

$$P_0(\lambda), P_1(\lambda), ..., P_n(\lambda) \tag{1}$$

обладает свойством последовательности Штурма: если $S(\mu)$ - число совпадение знаков в (1) у соседних элементов, при $\lambda = \mu$, то число собственных значений матрицы S_n, больших μ, совпадает с $S(\mu)$. Например, в последовательности 1,3,-3,5,2,0, -3,4,5,-3,-6 число совпадений знаков у соседних пар элементов равно 4. Ноль не рассматривается.

Все собственные числа лежат на отрезке [a,b]. Найдем k-ое собственное число. Ясно, что $S(a) > k$ и $S(b) < k$. Методом бисекции найдем более узкий интервал, содержащий корень λ_k. Для этого найдем середину интервала $c = (a + b)/2$. Подсчитаем $S(c)$, то есть число совпадения знаков в (1) при $\lambda = c$. Если $S(c) \geqslant n - k + 1$, то полагаем $a_1 = c$, $b_1 = b$. Если $S(c) < n - k + 1$, то полагаем $a_1 = a$, $b_1 = c$. Тогда $\lambda_k \in [a_1, b_1]$. Повторяем этот процесс до тех пор, пока $|b_t - a_t| < \varepsilon$. Тогда $\lambda_k \approx b_t$.

Найдем, например, второе собственное число трехдиагональной матрицы a из п.2. Введем функции, возвращающие матрицу главного минора $mg[k]$ порядка k матрицы a и ее характеристический многочлен $p[k]$. Определим многочлен $p[0] = 1$.

```
In[1]:= mg[k_] := a[[1;;k, 1;;k]]
        p[k_] := Det[mg[k] − λ IdentityMatrix[k]]
        p[0] = 1;
```

Составим список многочленов (1):

```
In[2]:= sp[λ_] = Table[p[i], {i, 0, 5}]
Out[2]= {1, 6.18749 − λ, λ^2 − 13.9338λ + 46.632, −λ^3 + 21.7511λ^2 − 152.686λ + 346.776,
          λ^4 − 28.918λ^3 + 307.423λ^2 − 1425.03λ + 2431.64,
          −λ^5 + 36.9985λ^4 − 541.074λ^3 + 3908.69λ^2 − 13943.2λ + 19641.2}
```

Тогда значения многочленов, например, при $\lambda = 6$ можно получить так

```
In[3]:= sp[6]
Out[3]= {1,0.18749,-0.970727,-2.30225,-1.56933,-3.21296}
```

Следующая команда $zn[\lambda]$ возвращает число совпадений знаков соседних элементов в списке значений многочленов при конкретном значении λ. Из списка значений сначала удаляются нули командой DeleteCases.

In[4]:=
```
zn[λ_] := Block[{j = 0, u},
                u = DeleteCases[sp[λ], 0];
                Do[
                 If[u[[i + 1]] u[[i]] > 0, j++],
                {i, 1, 5}]; j]
```

Например, в списке значений sp[6] число совпадений знаков соседних элементов равно 4:

In[5]:=
```
zn[6]
```
Out[5]= 4

Зададим теперь интервал, содержащий все собственные значения матрицы a :

In[6]:=
```
a1 = -nr[a]; b1 = nr[a];
```

Будем искать второе собственное значение методом бисекций. Пусть

In[7]:=
```
m = 2;
```

Найдем середину отрезка $[a1, b1]$:

In[8]:=
```
c1 = (a1 + b1)/2
```
Out[8]= 6.29219

Найдем число совпадений знаков в списке sp[c1] и, тем самым, найдем число собственных значений матрицы больших $c1$:

In[9]:=
```
zn[c1]
```
Out[9]= 3

Если $zn[c1] \geqslant 6 - m = 4$, то второе собственное значение будет принадлежать отрезку $[c1, b1]$, в противном случае - отрезку $[a1, c1]$. Следующая команда и отслеживает эти случаи. Команда Print приведена для удобства, она выводит изменяющийся конец интервала и его значение.

In[10]:=
```
If[zn[c1] >= 6 - m, Print["a1 = ", a1 = c1],
   Print["b1 = ", b1 = c1]]
```
Out[10]= b1= 6.29219

Проверяем длину интервала

In[11]:=
```
Abs[b1 - a1] < ε
```

Если результат действия этой команды будет False, то повторяем все команды с 8-11, пока 11 команда не вернет True.

Ясно, что повторяющиеся команды нужно объединить командой цикла

In[1]:=
```
bissob[f_, m_, {a1_, b1_}, ε_] :=
            Block[{a = a1, b = b1},
                While[Abs[b - a] > ε,
                c = (a + b)/2.;
                If[(c//f) >= 6 - m, a = c, b = c]
                ];
                c]
```

Здесь, вместо f нужно поместить zn без аргумента, m - порядковый номер собственного значения матрицы a, $\{a1, b1\}$ интервал, содержащий все собственные значения, а ε - точность вычислений.

Пример применения команды. Предполагается, что команда $zn[\lambda]$ определена, концы интервала $\{a1, b1\}$ введены:

In[2]:=
```
bissob[zn, 2, {a1, b1}, 0.0001]
```
Out[2]= 6.28825

n.5 Алгоритм QR-разложений

Алгоритм QR ([10]) применяется для решения полной проблемы собственных значений произвольной матрицы $A = (a_{i,j})$. Данный алгоритм является итерационным, каждый шаг итерации состоит в разложении матрица на произведение левой треугольной и ортогональной матриц. Исходная матрица приводится сначала к почти треугольному вида методом Гивенса, что сокращается число итераций алгоритма QR.

Алгоритм состоит в построении двух последовательностей матриц. Пусть $A_0 = A = (a_{i,j})$, строим разложения

$$\begin{gathered} A_0.Q_1 = \Lambda_1, \qquad Q_1^\top.\Lambda_1 = A_1, \\ A_1.Q_2 = \Lambda_2, \qquad Q_2^\top.\Lambda_2 = A_2, \\ \ldots \\ A_i.Q_{i+1} = \Lambda_{i+1}, \qquad Q_{i+1}^\top.\Lambda_{i+1} = A_{i+1}, \\ \ldots \end{gathered} \tag{1}$$

Здесь, $Q_i = T_{1,2}.T_{2,3}....T_{n-1,n}$, где $T_{i,i+1}$ - матрица вращения. Коэффициенты c и s матрицы вращения подбираются так, чтобы при умножении преобразуемой матрицы на матрицу вращения $T_{i,i+1}$ обнулялся элемент в позиции (i,i+1):

$$c = \frac{a_{i,i}}{\sqrt{a_{i,i}^2 + a_{i,i+1}^2}}, \qquad s = \frac{a_{i,i+1}}{\sqrt{a_{i,i}^2 + a_{i,i+1}^2}}$$

Рассмотрим более подробно процесс получения, например, матрицы Λ_1. Сначала матрица A_0 умножается справа на матрицу вращения $T_{1,2}$, коэффициенты которой определяются матрицей A_0. В результате получается вспомогательная матрица $A_{0,1}$. Матрица $A_{0,1}$ умножается справа на матрицу вращения $T_{2,3}$, коэффициенты которой определяются матрицей $A_{0,1}$. В результате получается вспомогательная матрица $A_{0,2}$. И так далее. Наконец, умножаем вспомогательную матрицу $A_{0,n-2}$ справа на матрицу вращения $T_{n-1,n}$. Произведение всех таких матриц вращения и дает матрицу Q_1. При умножении, матрицы Q_1 справа на матрицу почти треугольного вида A_0 происходит обнуление наддиагональных элементов, то есть результат умножения есть треугольная матрица Λ_1. В соответствии с правилом (1), определяем матрицу $A_1 = Q_1^\top.\Lambda_1$ и умножим ее справа на Q_2, которая получена как произведение матриц вращения, построенных с помощью матрицы A_1, так, как это сделано для матрицы A_0. В результате получаем снова треугольную матрицу Λ_2. Определяем матрицу $A_2 = Q_2^\top.\Lambda_2$ и так далее.

Из (1) получим

$$A_{i+1} = Q_{i+1}^\top.A_i.Q_{i+1}$$

или

$$A_{i+1} = (T_{n-1n}^\top.T_{n-2n-1}^\top....T_{12}^\top).A_i.(T_{12}.T_{23}....T_{n-1n}) \tag{2}$$

Процесс разложения матриц (1) останавливается, как только достигнуто такое значение i, при котором сумма произведений модулей соответствующих пар наддиагональных элементов матрицы A_i меньше наперед заданного ε:

$$\sum_{k=1}^{n-2} |a^{(i)}_{k,k+1} a^{(i)}_{k+1,k+2}| < \varepsilon \tag{3}$$

При выполнении требования (3), на главной диагонали матрицы $A_i = (a^{(i)}_{k,l})$ появятся клетки размерности 1 (диагональные элементы матрицы A_i) или клетки размерности 2 (миноры второго порядка). Как определить порядок клетки? Делается это так. Начнем рассматривать все клетки, начиная с левой верхней, для определения их размерности. Сначала смотрим на абсолютные значения элементов, стоящих в позициях (1,2) и (2,3). В силу (3) их произведение меньше ε. Если абсолютное значение элемента (1,2) меньше абсолютного значения элемента (2,3), то считаем, что в первой строке стоит клетка размерности 1. Это значит, что элемент $a^{(i)}_{1,1}$ является первым собственным значением матрицы a. Далее рассматриваем абсолютные значения элементов, стоящих в позициях (2,3) и (3,4) и так далее.

Если абсолютное значение элемента (1,2) больше абсолютного значения элемента (2,3) или равно, то считаем, что у нас на диагонали стоит клетка 2-го порядка:

$$\begin{pmatrix} a^{(i)}_{1,1} & a^{(i)}_{1,2} \\ a^{(i)}_{2,1} & a^{(i)}_{2,2} \end{pmatrix}$$

Решая характеристическое уравнение

$$\begin{vmatrix} a^{(i)}_{1,1} - \lambda & a^{(i)}_{1,2} \\ a^{(i)}_{2,1} & a^{(i)}_{2,2} - \lambda \end{vmatrix} = 0$$

находим либо два действительных, либо два комплексно-сопряженных собственных значения матрицы A. Если корни действительные, то упорядочим их по убыванию величин абсолютных значений и считаем, что найдены два собственных значения матрицы A. Далее рассматриваем абсолютные значения элементов, стоящих в позициях (3,4) и (4,5) и так далее.

В зависимости от размерности матрицы в конце этих процедур останется либо клетка 2-го порядка, с которой поступаем описанным выше способом, либо клетка 1-го порядка, элемент которой будет последним собственным значением матрицы A_i.

Доказано, что алгоритм QR разложения сходится, если собственные значения матрицы A, кроме комплексно-сопряженных собственных значений, различны по модулю.

Пример. Введем матрицу

In[1]:=
```
a = a1 = Table[Random[Integer, {5, 10}], {i, 1, 5}, {j, 1, 5}];
```

Приведем ее к почти треугольному виду методом Гивенса:

In[2]:=
```
Do[
    t1 = IdentityMatrix[5];
    c = a[[i - 1, i]]/Sqrt[a[[i - 1, i]]^2 + a[[i - 1, j]]^2]//N;
    s = a[[i - 1, j]]/Sqrt[a[[i - 1, i]]^2 + a[[i - 1, j]]^2]//N;
    t1[[i, i]] = t1[[j, j]] = c;
    t1[[i, j]] = -s;
    t1[[j, i]] = s;
```

```
    a = Transpose[t1].a.t1//Chop//N,
  {i, 2, 4}, {j, i + 1, 5}]
```

Выведем матрицу a и сделаем ее дубль:

In[3]:=
```
a1 = a
```

Out[3]=

$$\begin{pmatrix} 13 & 20.63976 & 0 & 0 & 0 \\ 25.38788 & 32.62441 & 19.70813 & 0 & 0 \\ 10.39849 & 17.41909 & 3.17013 & 1.74163 & 0 \\ 2.99418 & 2.02566 & -1.84224 & -1.12864 & 4.64103 \\ 1.16689 & 7.97588 & 0.89451 & -4.26951 & -1.66590 \end{pmatrix}$$

Введем матрицу вращений построенную по матрице a :

In[4]:=
```
t[a_, i_, j_] := Block[{c, s},
                  t1 = IdentityMatrix[5];
                  c = a[[i, i]]/Sqrt[a[[i, i]]^2 + a[[i, j]]^2]//N;
                  s = a[[i, j]]/Sqrt[a[[i, i]]^2 + a[[i, j]]^2]//N;
                  t1[[i, i]] = t1[[j, j]] = c;
                  t1[[i, j]] = -s;
                  t1[[j, i]] = s; t1]
```

Составим цикл, остановка которого будет при выполнении условия (3). Значением f будет число проходов цикла. Умножения (1) левого столбца выполняются последовательно командами $a = a.tq$, где матрица $tq = t[a, i, i + 1]$ вычисляется всякий раз по преобразованной вспомогательной матрице a. Произведения правого столбца в (1) осуществляются командой $a = Transpose[q].a$, где матрица q есть произведение соответствующих матриц вращения tq.

In[5]:=
```
f = 0;
While[
  Sum_{i=1}^{3} Abs[a[[i, i + 1]] a[[i + 1, i + 2]]] > 0.01,
  f = f + 1;
  q = IdentityMatrix[5];
  Do[
    tq = t[a, i, i + 1];
    q = q.tq;
    a = a.tq//Chop,
  {i, 1, 4}];
  a = Transpose[q].a;
];
f
```

Цикл проработает 25 раз:

Out[5]= 25

В результате получим матрицу

In[6]:=
```
a//Chop
```

Out[6]=

$$\begin{pmatrix} 54.1502 & 0 & 0 & 0 & 0 \\ 1.17848 & -7.42172 & -0.00186 & 0 & 0 \\ 0.8456 & -1.92113 & -1.69913 & -5.35968 & 0 \\ 1.3679 & -5.3455 & 5.4477 & -0.87512 & 4.287 \times 10^{-10} \\ 11.12375 & 4.87310 & -0.75992 & -2.42984 & 1.84576 \end{pmatrix}$$

Следуем алгоритму распознования клеток. Так как нулевой элемент в позиции (1,2) меньше абсолютного значения элемента -0.00186, стоящего в позиции (2,3), то элемент 54.1502 образует клетку размерности 1 и является первым собственным значением матрица a. Абсолютное значение элемента -0.00186 меньше абсолютного значения элемента -5.35968, стоящего в позиции (3,4), поэтому диагональный элемент -7.42172 является вторым собственным значением матрицы a. Элемент -5.35968, стоящий в позиции (3,4), по абсолютной величине больше элемента 4.287×10^{-10}. Поэтому на диагонали стоит клетка 2-го порядка

$$\begin{pmatrix} -1.69913 & -5.35968 \\ 5.4477 & -0.87512 \end{pmatrix}$$

Решаем ее характеристическое уравнение:

In[7]:= $\mathbf{i = 3}$;

In[8]:= $\mathbf{Solve[(a[[i,i]]-x)(a[[i+1,i+1]]-x)-a[[i,i+1]]a[[i+1,i]]==0,x]}$

Out[8]= $\{\{x\to -1.28713-5.3878\,i\},\{x\to -1.28713+5.3878\,i\}\}$

Таким образом, третье и четвертое собственные значения являются комплексными. Остался пятый диагональный элемент 1.84576 - последнее действительное собственное число матрицы a.

В результате получаем следующие собственные значения матрицы a:
$\{54.1502, -7.42172, -1.28713+5.3878\,i, -1.28713-5.3878\,i, 1.84576\}$

Для сравнения приведем решение задачи встроенной функцией

In[9]:= $\mathbf{Eigenvalues[a1]//N}$

Out[9]= $\{54.1502, -7.42288, -1.28656+5.38812\,i, -1.28656-5.38812\,i,$
$1.84577\}$

Процесс определения собственных значений из клеток матрицы a можно механизировать. Введем две команды $vsp[a,i]$ и $korni[a]$. Первая команда $vsp[a,i]$ решает характеристическое уравнение i-ой двумерной клетки, упорядочивает решение по убыванию абсолютных значений и присоединяет найденные собственные значения к списку уже найденных собственных значений rt. Вторая команда $korni[a]$ сравнивает абсолютные значения элементов, стоящих в позициях (i,i+1) и (i+1,i+2) и присоединяет к списку уже найденных собственных значений rt или к пустому первоначальному списку $rt=\{\}$ либо соответствующий диагональный элемент $a[[i,i]]$, либо вызывается команда $vsp[a,i]$ и к списку rt добавляется пара собственный значений. Значение i первоначально равно 1 и в цикле, каждый раз, увеличивается на 1 или 2 в зависимости от размерности рассмотренной в данный момент клетки. Цикл останавливается, как только $i \geqslant Length[a]$. При этом, в цикле могут остаться не рассмотренными последние диагональные клетки размерности 1 или 2. Эти случаи рассматриваются отдельно в двух последних командах. Команда $korni[a]$ возвращает список собственных значений матрицы a.

Для применения этих команд их надо ввести один раз:

In[7]:=
```
vsp[a_,i_]:=Block[{q,d},
    q=Solve[(a[[i,i]]-x)(a[[i+1,i+1]]-x)-
        a[[i,i+1]]a[[i+1,i]]==0,x];
    d=If[Abs[q[[2,1,2]]]<=Abs[q[[1,1,2]]],x/.q,Reverse[x/.q]];
        rt=Append[rt,d]//Flatten;]
```

In[8]:=
```
korni[a_]:=Block[{rt={},i=1,q,d},
```

```
While[i < Length[a] - 1,
  If[Abs[a[[i, i + 1]]] ⩽ Abs[a[[i + 1, i + 2]]],
    rt = Append[rt, a[[i, i]]]//Flatten; i = i + 1,
    vsp[a, i]; i = i + 2]
];
If[i == Length[a] - 1, vsp[a, i]];
If[i == Length[a], rt = Append[rt, a[[n, n]]]]; rt]
```

и выполнить команду

```
In[9]:= korni[a]
Out[9]= {54.1502, -7.42172, -1.28713 + 5.3878 i, -1.28713 - 5.3878 i,
  1.84576}
```

n.6 Собственные значения симметричной матрицы. Метод Якоби

Найдем собственные значения симметрической матрицы $A = (a_{ij})$, $i, j = 1, ..., n$, методом Якоби. Предварительно, матрицу A можно привести к трехдиагональному виду с помощью метода Гивенса (§12, n.2), что позволит ускорить решение задачи.

Рассмотрим сначала две алгебраические леммы. Напомним определение: матрицы A и B подобны, если существует невырожденная матрица C такая, что $B = C^{-1}.A.C$.

Лемма 1. Пусть $\{\lambda, x\}$ собственная пара матрицы $B = C^{-1}.A.C$, тогда $\{\lambda, C.x\}$ - собственная пара матрицы A.

Доказательство. Если $B.x = \lambda x$, то $(C^{-1}.A.C).x = \lambda x$. Отсюда $C.(C^{-1}.A.C).x = C.(\lambda x)$ или $A.(C.x) = \lambda(C.x)$.

Лемма 2. Если диагональная матрица Λ и матрица A подобны

$$\Lambda = X^{-1}.A.X, \tag{1}$$

то диагональные элементы Λ есть собственные числа, а столбцы матрицы X есть собственные векторы матрицы A.

Доказательство. Пусть λ_i есть i - ый диагональный элемент матрицы Λ. Собственный вектор, соответствующий элементу λ_i, есть i - ый базисный вектор $e_i = (0, ..., 1, ..., 0)^\top$. Это следует из легко проверяемого равенства $\Lambda.e_i = \lambda_i e_i$. По свойству 1, $\{\lambda_i, X.e_i\}$ есть собственная пара матрицы A. Но $X.e_i = x_i$ - i-ый столбец матрицы X. Поэтому $\{\lambda_i, x_i\}$ - собственная пара матрицы A.

Пусть теперь матрица A - симметричная. Известно, что у такой матрицы все собственные значения действительны, а собственные векторы образуют ортонормированную систему векторов. Если столбцы матрицы X есть собственные векторы симметричной матрицы, то матрица X - ортогональная матрица. Так как для ортогональной матрицы справедливо равенство $X^{-1} = X^\top$, то для симметричной матрицы A соотношение (1) примет вид

$$\Lambda = X^\top.A.X, \tag{2}$$

где диагональные элементы Λ есть собственные значения матрицы A.

Метод Якоби заключается в приближенной реализации равенства (2). Заданная погрешность приближения получается в результате последовательности однотипных преобразований. Опишем первое такое преобразование.

Пусть a_{ij} есть максимальный по модулю недиагональный элемент матрицы A. Считаем, что $a_{ij} \neq 0$, в противном случае A - диагональная матрица. Построим матрицу вращения T_{ij} (§6) равного с A порядка:

$$T_{ij} = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & c & \dots & s & & \\ & & & \ddots & & & \\ & & -s & \dots & c & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}$$

где $s^2 + c^2 = 1$, элемент c стоит в позициях (i,i) и (j,j), элемент s в позиции (i,j), а $-s$ в позиции (j,i). Матрица T_{ij} ортогональная, поэтому матрица

$$B = T_{ij}^{\top}.A.T_{ij} \tag{3}$$

подобна матрице A и, значит, имеет тот же спектр. Элементы c и s подбираем так, чтобы элемент матрицы B, стоящий в позиции (i,j), стал нулевым.

Второе преобразование совершается над матрицей B точно также, находим максимальный по модулю недиагональный элемент матрицы B и обнуляем его подобным образом. И так далее. В результате получаем последовательность матриц $B_0 = A$, $B_1 = B$, ..., B_k, ..., таких, что

$$B_k = T_{i_k j_k}^{\top}.B_{k-1}.T_{i_k j_k}, \quad i_1 = i, \quad j_1 = j, \quad k = 1, 2, ... \tag{4}$$

или, подставляя в (4) вместо матрицы B_{k-1} ее выражение через B_{k-2} по формуле (4) и так далее, получим

$$B_k = (T_{i_1 j_1}.T_{i_2 j_2}...T_{i_k j_k})^{\top}.A.(T_{i_1 j_1}.T_{i_2 j_2}...T_{i_k j_k}), \quad k = 1, 2, ... \tag{5}$$

Оказывается, что при $k \to \infty$ матрица B_k стремится к диагональной матрице, соответственно равенство (4) приближенно реализует равенство (3). Как только сумма квадратов недиагональных элементов матрицы (4) станет меньше наперед заданного $\varepsilon > 0$, то процесс построения матриц (4) останавливается - диагональные элементы матрицы B_k будут приближениями собственных значений матрицы A. Столбцы матрицы $Q = T_{i_1 j_1}.T_{i_2 j_2}...T_{i_k j_k}$ будут собственными векторами матрицы A.

Вычисление элементов s и c. Рассмотрим первое преобразование, то есть переход от матрицы A к матрице $B = (b_{ij})$. По свойству матриц вращения (§6) матрица B отличается от матрицы A только i, j строками и столбцами. Перемножая матрицы в (3) получим, что при $m \neq i, j$ элементы i, j строк и столбцов преобразуются по формулам

$$b_{im} = b_{mi} = ca_{mi} + sa_{mj}, \quad b_{jm} = b_{mj} = -sa_{mi} + ca_{mj}, \tag{6}$$

а элементы, стоящие на пересечении i, j строк и столбцов, можно вычислить так

$$b_{ii} = c^2 a_{ii} + s^2 a_{jj} + 2sca_{ij}, \quad b_{jj} = c^2 a_{jj} + s^2 a_{ii} - 2sca_{ij},$$

$$b_{ij} = b_{ji} = c^2 a_{ij} - sca_{ii} + sca_{jj} - s^2 a_{jj}.$$

Все остальные элементы матриц B и A совпадают.

Требуем, чтобы $b_{ij} = b_{ji} = 0$, то есть $c^2 a_{ij} - sca_{ii} + sca_{jj} - s^2 a_{jj} = 0$ или

$$(c^2 - s^2)a_{ij} - sc(a_{ii} - a_{jj}) = 0. \tag{7}$$

Так как $s^2 + c^2 = 1$, то можно считать, что $c = \cos\alpha$, $s = \sin\alpha$, где α принадлежит отрезку $[-\frac{\pi}{4}, \frac{\pi}{4}]$. Тогда (7) примет вид $(\cos^2\alpha - \sin^2\alpha)a_{ij} - \cos\alpha\sin\alpha(a_{ii} - a_{jj}) = 0$ или

$$\cos(2\alpha)a_{ij} - \frac{1}{2}\sin(2\alpha)(a_{ii} - a_{jj}) = 0. \tag{8}$$

Введем обозначение $p = 2a_{i,j}$, $q = a_{i,i} - a_{j,j}$. Если $q = 0$, то из (7) получаем $c = s$. Следовательно, в этом случае $\cos\alpha = \sin\alpha$, $\alpha = \frac{\pi}{4}$, а $c = s = \frac{\sqrt{2}}{2}$.

Если $q \neq 0$, то из (8) получим

$$\tan 2\alpha = \frac{p}{q}.$$

Отсюда

$$\cos(2\alpha) = \frac{1}{\sqrt{1 + \tan^2(2\alpha)}} = \frac{1}{\sqrt{1 + (\frac{p}{q})^2}} = \frac{|p|}{\sqrt{p^2 + q^2}}.$$

Введем обозначение: $d = \sqrt{p^2 + q^2}$, $r = \frac{|q|}{2d}$. Тогда

$$c = \cos\alpha = \sqrt{\frac{1}{2}(1 + \cos(2\alpha))} = \sqrt{\frac{1}{2} + \frac{|q|}{2d}} = \sqrt{0.5 + r},$$

а

$$s = \pm\sqrt{1 - \cos^2\alpha} = \pm\sqrt{0.5 - r} = \sqrt{0.5 - r}\,\mathrm{sign}(p \cdot q),$$

так как знак синуса для искомых углов совпадает со знаком тангенса угла 2α.

Таким образом, одно преобразование (первое) метода Якоби требует следующих вычислений:

$$\begin{aligned} &p = 2a_{i,j},\\ &q = a_{i,i} - a_{j,j},\\ &d = \sqrt{p^2 + q^2},\\ &r = |q|/(2d),\\ &c = \sqrt{0.5 + r},\\ &s = \sqrt{0.5 - r}\,\mathrm{sign}(p \cdot q). \end{aligned}$$

Если $q = 0$, то $s = c = \frac{\sqrt{2}}{2}$. Формируем матрицу вращения T_{ij} и по формуле (3) строим матрицу B. Матрицу B также можно получить, изменяя i, j строки и столбцы матрицы A, по приведенным выше формулам. Далее, повторяем аналогичные вычисления для матрицы B и так далее.

Оценка суммы квадратов недиагональных элементов. Рассмотрим евклидову норму $||A|| = \sqrt{\sum_{i,j=1}^{n} a_{ij}^2}$ матрицы $A = (a_{ij})$. Обозначим через $\overline{A}$ матрицу, полученную из матрицы A обнулением элементов её главной диагонали. Матрицы $\overline{A}$ и A отличаются только диагональными элементами. Покажем, что сумма квадратов недиагональных элементов при преобразовании матриц по формуле (4) не возрастает. Рассмотрим первое преобразование, то есть переход от матрицы A к матрице B. Так

как эти матрицы различаются только элементами i, j строк и столбцов, то надо подсчитать сумму квадратов элементов i, j строк и столбцов матрицы B. Применяя (6), найдем сумму квадратов недиагональных элементов i строки и j столбца матрицы B. При $m \neq i, j$ запишем

$$\sum_{m=1}^{n} b_{im}^2 + \sum_{m=1}^{n} b_{mj}^2 = \sum_{m=1}^{n} (b_{im}^2 + b_{mj}^2) = \sum_{m=1}^{n} ((ca_{im} + sa_{mj})^2 + (-sa_{im} + ca_{mj})^2) =$$

$$\sum_{m=1}^{n} (a_{im}^2 (c^2 + s^2) + a_{mj}^2 (c^2 + s^2)) = \sum_{m=1}^{n} (a_{im}^2 + a_{mj}^2).$$

Сумма квадратов недиагональных элементов, стоящих в i строке и j столбце матрицы A равна

$$\sum_{m=1}^{n} (a_{im}^2 + a_{mj}^2) + a_{ij}^2.$$

Видим, что сумма квадратов недиагональных элементов, стоящих в i строке и j столбце матриц A и B отличаются на a_{ij}^2. Аналогично получаем, что сумма квадратов недиагональных элементов, стоящих в j строке и i столбце матриц A и B отличаются также на a_{ij}^2. Отсюда получаем, что

$$||\overline{B}||^2 = ||\overline{A}||^2 - 2a_{ij}^2. \tag{9}$$

Если a_{ij} максимальный по модулю недиагональный элемент матрицы A, то его квадрат не меньше среднего значения квадратов недиагональных элементов. Так как недиагональных элементов $n^2 - n$, то можно записать, что

$$a_{ij}^2 \geqslant \frac{1}{n(n-1)} \sum_{m \neq l} a_{ml}^2 = \frac{1}{n(n-1)} ||\overline{A}||^2.$$

Отсюда и из (9) получим, что

$$||\overline{B}||^2 \leqslant ||\overline{A}||^2 - \frac{2}{n(n-1)} ||\overline{A}||^2 = (1 - \frac{2}{n(n-1)}) ||\overline{A}||^2. \tag{10}$$

Применяя (10) для матриц последовательности (4), получим

$$||\overline{B}_k||^2 \leqslant (1 - \frac{2}{n(n-1)}) ||\overline{B}_{k-1}||^2 \leqslant (1 - \frac{2}{n(n-1)})^2 ||\overline{B}_{k-2}||^2 \leqslant$$

$$\leqslant ... \leqslant (1 - \frac{2}{n(n-1)})^k ||\overline{A}||^2.$$

Так как $1 - \frac{2}{n(n-1)} < 1$, то при $k \to \infty$ сумма квадратов недиагональных элементов $||\overline{B}_k|| \to 0$.

Формулировка задания. Введем евклидову норму матрицы:

In[1]:= $\mathbf{nr}[\mathbf{t\_}] := \mathbf{Sqrt}[(\mathbf{t}//\mathbf{Flatten}).(\mathbf{t}//\mathbf{Flatten})]$

Сгенерируем матрицу

In[2]:= $(\mathbf{m} = \mathbf{Table}[\mathbf{Random}[\mathbf{Integer}, \{-\mathbf{20}, \mathbf{40}\}],$

$\{\mathbf{i},\mathbf{1},\mathbf{5}\},\{\mathbf{j},\mathbf{1},\mathbf{5}\}]]//\mathbf{MatrixForm}$

Out[2]=

$$\begin{pmatrix} 30 & -3 & 37 & -3 & 27 \\ -12 & 33 & 4 & 1 & -12 \\ 21 & 4 & 2 & 28 & -12 \\ 37 & 30 & 11 & 31 & 37 \\ 26 & 3 & 15 & -5 & 11 \end{pmatrix}$$

и сделаем из нее симметрическую матрицу

In[3]:= $\mathbf{Do}[\mathbf{If}[\mathbf{i}>\mathbf{j},\mathbf{m}[[\mathbf{i},\mathbf{j}]]=\mathbf{m}[[\mathbf{j},\mathbf{i}]]],\{\mathbf{i},\mathbf{1},\mathbf{5}\},\{\mathbf{j},\mathbf{1},\mathbf{5}\}]$

Out[3]=

$$\begin{pmatrix} 30 & -3 & 37 & -3 & 27 \\ -3 & 33 & 4 & 1 & -12 \\ 37 & 4 & 2 & 28 & -12 \\ -3 & 1 & 28 & 31 & 37 \\ 27 & -12 & -12 & 37 & 11 \end{pmatrix}$$

Найдем максимальный по модулю недиагональный элемент симметричной матрицы m, построим матрицу вращения и обнулим найденный элемент в матрице m.

Построим вспомогательную диагональную матрицу

In[4]:= $\mathbf{dm}=\mathbf{DiagonalMatrix}[\mathbf{Table}[\mathbf{Abs}[\mathbf{m}[[\mathbf{i},\mathbf{i}]]],\{\mathbf{i},\mathbf{1},\mathbf{Length}[\mathbf{m}]\}]]$

Out[4]=

$$\begin{pmatrix} 30 & 0 & 0 & 0 & 0 \\ 0 & 33 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 31 & 0 \\ 0 & 0 & 0 & 0 & 11 \end{pmatrix}$$

Рассмотрим матрицу, состоящую из абсолютных значений элементов матрицы **m** у которой диагональные элементы равны нулю

In[5]:= $\mathbf{Abs}[\mathbf{m}]-\mathbf{dm}//\mathbf{MatrixForm}$

Out[5]=

$$\begin{pmatrix} 0 & 3 & 37 & 3 & 27 \\ 3 & 0 & 4 & 1 & 12 \\ 37 & 4 & 0 & 28 & 12 \\ 3 & 1 & 11 & 0 & 37 \\ 27 & 12 & 12 & 5 & 0 \end{pmatrix}$$

и найдем максимальный недиагональный элемент этой матрицы:

In[6]:= $\mathbf{e}=\mathbf{Max}[\mathbf{Abs}[\mathbf{m}]-\mathbf{dm}]$

Out[6]= 37

Находим позицию найденного элемента

In[7]:= $\{\mathbf{i},\mathbf{j}\}=\mathbf{First}[\mathbf{Position}[\mathbf{Abs}[\mathbf{m}]-\mathbf{dm},\mathbf{e}]]$

Out[7]= {1, 3}

Вводим единичную матрицу

In[8]:= $\mathbf{t}=\mathbf{IdentityMatrix}[\mathbf{Length}[\mathbf{m}]]$

Out[8]=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

и найдем элементы матрицы вращения. Вычислим последовательно величины

```
In[9]:= p = 2m[[i, j]];
        q = m[[i, i]] - m[[j, j]];
        If[q == 0,
          s = c = Sqrt[2.]/2,
          d = Sqrt[p^2 + q^2]; r = Abs[q]/(2d);
          c = Sqrt[0.5 + r]; s = Sqrt[0.5 - r]Sign[p q]];
```

Меняем в единичной матрице четыре элемента

```
In[10]:= t[[i, j]] = -s;
         t[[j, i]] = s;
         t[[i, i]] = t[[j, j]] = c;
```

В результате получим матрицу вращения

```
In[11]:= t
```

Out[11]=

$$\begin{pmatrix} 0.82276 & 0 & -0.56837 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0.56837 & 0 & 0.82276 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Обнуляем элемент матрицы $\mathbf{m}$ в позиции $\{\mathbf{i},\mathbf{j}\} = \{\mathbf{1},\mathbf{3}\}$:

```
In[12]:= (m = Transpose[t].m.t)//MatrixForm//Chop
```

Out[12]=

$$\begin{pmatrix} 55.5601 & -0.194789 & 0 & 13.4463 & 15.3942 \\ -0.194789 & 33. & 4.9962 & 1. & -12. \\ 0 & 4.9962 & -23.5601 & 24.7426 & -25.2194 \\ 13.4463 & 1. & 24.7426 & 31. & 37. \\ 15.3942 & -12. & -25.2194 & 37. & 11. \end{pmatrix}$$

Если в качестве оценки погрешности взять 0.001, то полученная матрица не будет иметь корень из суммы квадратов недиагональных элементов меньше 0.001. Поэтому все команды нужно повторит для вновь полученной матрицы m. Воспользуемся командой цикла и выполним следующие команды. Матрица *vec* является произведением матриц вращения полученных до данной итерации включительно.

```
In[13]:= vec = IdentityMatrix[Length[m]];
         Do[
          dm = DiagonalMatrix[Table[Abs[m[[i, i]]],
            {i, 1, Length[m]}]];
          e = Max[Abs[m] - dm];
          {i, j} = First[Position[Abs[m], e]];
          q = m[[i, i]] - m[[j, j]];
          If[q == 0,
            c = s = Sqrt[2.]/2,
            p = 2m[[i, j]];
```

```
    d = Sqrt[p^2 + q^2];
    r = Abs[q]/(2d);
    c = Sqrt[0.5 + r];
    s = Sqrt[0.5 - r]Sign[p q];
  ];
  t = IdentityMatrix[Length[m]];
  t[[i, j]] = -s;
  t[[j, i]] = s;
  t[[i, i]] = t[[j, j]] = c;
  m = Transpose[t].m.t;
  vec = vec.t;
  If[nr[Abs[m] - dm] < 0.001,
  Print["На", k, "шаге матрица m = ",
  m//MatrixForm//Chop,
   "Собственные значения : ", Table[m[[u, u]],
  {u, 1, Length[m]}]];
  Break[]],
 {k, 1, 100}]
```

Out[13]=

На 25 шаге матрица m=

$$\begin{pmatrix} 40.0368 & 0 & -0.00002 & -0.0001 & 6.92662\times10^{-8} \\ 0 & 35.9941 & 3.8212\times10^{-8} & 0.00008 & 0.00005 \\ -0.00002 & 3.8212\times10^{-8} & -55.8829 & 5.6324\times10^{-9} & 0.00003 \\ -0.0001 & 0.00008 & 5.6324\times10^{-9} & 78.2825 & 0 \end{pmatrix}$$

Собственные значения: {40.0368, 35.9941, -55.8829, 78.2825, 8.56963}

Собственные векторы соответствующие данным собственным значениям являются строками матрицы

In[14]:=
```
Transpose[vec]
```

Out[14]=

$$\begin{pmatrix} 0.459706 & 0.629588 & 0.420901 & -0.283096 & -0.367407 \\ -0.501684 & 0.636268 & 0.00524088 & 0.585791 & 0.0172288 \\ -0.433832 & 0.0344717 & 0.591689 & -0.429726 & 0.525206 \\ 0.53574 & -0.115594 & 0.386607 & 0.566846 & 0.478373 \\ 0.248505 & 0.429222 & -0.568555 & -0.265943 & 0.600028 \end{pmatrix}$$

Глава 2

Численные методы математического анализа

§13 Численное решение нелинейных уравнений

В этом параграфе рассматриваются методы решения уравнения вида $f(x) = 0$: метод итераций, метод Ньютона и метод деления отрезка пополам (метод бисекций). Применение методов рассматривается на конкретном примере.

n.1 Изоляция корня уравнения

Применение метода, например, такого как деление отрезка пополам, предполагает, что на рассматриваемом отрезке есть корень уравнения. Поэтому вначале надо определить такие отрезки, то есть провести изоляцию корня уравнения.

Изоляция корня уравнения основывается на следствии из теоремы Коши: если для непрерывной функции $f(x)$ выполняется условие $f(a)f(b) < 0$, то на отрезке $[a, b]$ есть по крайней мере один корень уравнения $f(x) = 0$ и есть точно один корень, если функция монотонная.

Для изоляции корня можно поступить следующим образом. Интервал $[a, b]$, на котором определена функция $f(x) = 0$, разбивается точками $a = x_0 < x_1 < ... < x_n = b$ на равные отрезки длины h. Если $f(x_i)f(x_{i+1}) < 0$, то на отрезке $[x_i, x_{i+1}]$ есть корень данного уравнения. Если таких отрезков нет, то следует уменьшить шаг h, например, вдвое и повторить проверку условия.

В примерах ниже используется графический метод изоляции корня. Строится график функции $f(x)$ и определяется интервал, содержащий точку пересечения графика и оси абсцисс.

Еще один способ основан на рассмотрении мажорант функции $f(x)$: если a и b нули функций $p(x)$ и $q(x)$ таких, что $p(x) \leqslant f(x) \leqslant q(x)$, то ноль функции $f(x)$ принадлежит отрезку $[min(a, b), max(a, b)]$.

Сформулируем задачу. Требуется найти корень уравнения

$$\mathbf{2^x - x - 10 = 0} \tag{1}$$

с точность

In[1]:= $\mathbf{e = 0.0001}$

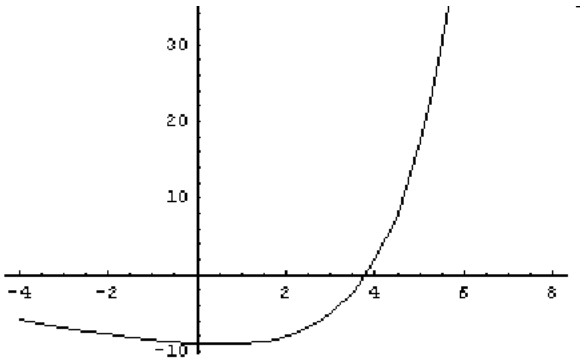
Проведем изоляцию корня. Введем функцию:

In[2]:= $\mathbf{f[x\_] := 2^x - x - 10}$

и построим ее график. Интервал, на котором строится график, выбирается вначале произвольно, затем изменяется так, чтобы ясно был виден интервал изоляции корня. Отметим, что в программе Mathematica оси координат могут пересекаться не в начале координат, обычно оси координат смещены в область графика. При произвольном расположении осей координат можно ошибиться в определении интервала изоляции корня и в приближенном определении корня уравнения. Опция $AxesOrigin \to \{0,0\}$ в команде *Plot* указывает координаты точки пересечения осей координат.

In[3]:= $\mathbf{Plot[f[x], \{x, -4, 8\},\ AxesOrigin \to \{0,0\}]}$

Out[3]=



Построенный график позволяет провести изоляцию корней данного уравнения. На отрезке [a,b], где a=2, b=5 находится единственный корень уравнения, приближенно равный 4. Вводим концы отрезка:

In[4]:= $\mathbf{a = 2;}$

$\mathbf{b = 5;}$

Решим данное уравнение тремя способами - методом итераций, методом Ньютона и методом половинного деления.

n.2 Метод итераций

Приведем левую часть уравнения $f(x) = 0$ к виду $f(x) = x - \varphi(x)$, тогда уравнение можно записать в виде

$$x = \varphi(x). \tag{2}$$

Построим последовательность

$$x_n = \varphi(x_{n-1}), \quad n = 0, 1, 2, ..., \tag{3}$$

начиная с некоторого начального приближения x_0. Если полученная последовательность имеет предел ξ, функция $\varphi(x)$ непрерывна, то ξ есть решение уравнения (2). Действительно, достаточно перейти к пределу в (3) при $n \to \infty$:

$$\xi = \lim_{n\to\infty} x_n = \lim_{n\to\infty} \varphi(x_{n-1}) = \varphi(\lim_{n\to\infty} x_{n-1}) = \varphi(\xi).$$

Нахождение элемента последовательности $\{x_n\}$ такого, что

$$|\xi - x_n| < \varepsilon$$

составляет суть метода итерации. При этом, найденный элемент x_n, будет являться корнем данного уравнения с заданной точностью (абсолютной погрешностью) ε.

Теорема 1. Если $\varphi(x)$ непрерывно дифференцируемая функция на отрезке $[a,b]$,

$$|\varphi'(x)| \leqslant q < 1 \tag{4}$$

для всех $x \in [a,b]$ при некотором q, то последовательность $\{x_n\}$, построенная по формуле (3) начиная с произвольного начального приближения $x_0 \in [a,b]$ и такая, что $\{x_n\} \subset [a,b]$, сходится к единственному корню ξ уравнения (2).

Доказательство. Из определения последовательности $\{x_n\}$ и формулы Лагранжа, следует, что

$$|x_{n+1} - x_n| = |\varphi(x_n) - \varphi(x_{n-1})| = |\varphi'(\overline{x})||(x_n - x_{n-1}| \leqslant q|(x_n - x_{n-1}|. \tag{5}$$

Отсюда, при $n = 1 : |x_2 - x_1| \leqslant q|x_1 - x_0|$, при $n = 2 : |x_3 - x_2| \leqslant q|x_2 - x_1| \leqslant q^2|x_1 - x_0|$ и так далее, $|x_{n+1} - x_n| \leqslant q^n|x_1 - x_0|$. Поэтому, при любых натуральных n, p можно записать
$|x_{n+p} - x_n| = |x_{n+p} - x_{n+p-1} + x_{n+p-1} - x_{n+p-2} + ... + x_{n+1} - x_n| \leqslant |x_{n+p} - x_{n+p-1}| + |x_{n+p-1} - x_{n+p-2}| + ... + |x_{n+1} - x_n| \leqslant q^{n+p-1}|x_1 - x_0| + q^{n+p-2}|x_1 - x_0| + ... + q^n|x_1 - x_0| \leqslant q^n|x_1 - x_0|(q^{p-1} + q^{p-2} + ... + 1) < q^n|x_1 - x_0|(1 + .. + q^{p-2} + q^{p-1} + q^p + q^{p+1} + ...)$.

Здесь, к сумме в скобках добавили бесконечную последовательность $q^p + q^{p+1} + ...$ В скобках теперь стоит геометрическая прогрессия с первым элементом равным 1 и знаменателем прогрессии $q < 1$. Так как сумма такой геометрической прогрессии равна $\frac{1}{1-q}$, то окончательно получаем

$$|x_{n+p} - x_n| \leqslant \frac{q^n}{1-q}|x_1 - x_0|.$$

Отсюда следует, что последовательность $\{x_n\}$ является фундаментальной последовательностью и так как последовательность ограничена, то она имеет предел. Как отмечалось выше, этот предел будет и решением уравнения (2).

Допустим, что уравнение (2) имеет два решения на отрезке $[a,b] : \xi = \varphi(\xi)$ $\overline{\xi} = \varphi(\overline{\xi})$. Отсюда, применяя формулу Лагранжа, получим $|\xi - \overline{\xi}| = |\varphi(\xi) - \varphi(\overline{\xi})| = |\varphi'(c)||(\xi - \overline{\xi})|$. Следовательно $|\xi - \overline{\xi}|(1 - |\varphi'(c)|) = 0$. Так как $|\varphi'(c)| < 1$, то отсюда получаем, что $|\xi - \overline{\xi}| = 0$ или $\xi = \overline{\xi}$.

Оценка погрешности. Оценим модуль абсолютной погрешности $|x_n - \xi|$, где ξ - корень уравнения (2) или уравнения $f(x) = 0$, $\{x_n\}$ - последовательность, о которой говориться в условии теоремы. Так как $f(x) = x - \varphi(x)$, то $f'(x) = 1 - \varphi'(x) \geqslant 1 - q$ в силу оценки $|\varphi'(x)| \leqslant q$. Отсюда

$$|x_{n+1} - x_n| = |\varphi(x_n) - x_n| = |f(x_n)| = |f(x_n) - f(\xi)| = |f'(c)||x_n - \xi| \geqslant (1-q)|x_n - \xi|,$$

следовательно

$$|x_n - \xi| \leqslant \frac{1}{1-q}|x_{n+1} - x_n|,$$

а, применяя неравенство (5), окончательно получим

$$|x_n - \xi| \leqslant \frac{q}{1-q}|x_n - x_{n-1}| \tag{6}$$

или

$$|\xi - x_n| \leqslant |x_n - x_{n-1}|,$$

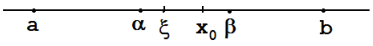
если $q \leqslant \frac{1}{2}$.

Такая оценка позволяет при заданной точности ε найти первый член последовательности $\{x_n\}$ такой, что $|\xi - x_n| < \varepsilon$. По формуле (3) строим элементы последовательности и на каждом шаге проверяем выполнение неравенства $\frac{q}{1-q}|x_n - x_{n-1}| < \varepsilon$. Первый член, удовлетворяющий такому требованию в силу оценки (6) и будет являться приближенным значением корня уравнения (2) с заданной точностью.

Число m называется порядком сходимости итерационной последовательности $\{x_n\}$ к решению ξ если $|x_k - \xi| < C|x_k - x_{k-1}|^m$. При m = 1 говорят о линейной сходимости; при m = 2 о квадратичной. Оценка (6) говорит о линейной сходимости итерационной последовательности к решению.

Отметим, что проверку условия $\{x_n\} \subset [a, b]$ на каждом шаге построения итерационной последовательности, о котором говориться в условии теоремы 1, можно избежать, если воспользоваться более практичной следующей теоремой.

Теорема 2. Пусть $\varphi(x)$ непрерывно дифференцируемая функция на отрезке $[a, b]$ и $|\varphi'(x)| \leqslant q < 1$ для все $x \in [a, b]$ и при некотором q. Последовательность $\{x_n\}$, построенная по формуле (3) принадлежит отрезку $[a, b]$, если корень ξ уравнения (2) и начальное приближение x_0 принадлежат более узкому отрезку $[\alpha, \beta]$, где $\alpha = a + \frac{b-a}{3}$, $\beta = b - \frac{b-a}{3}$.

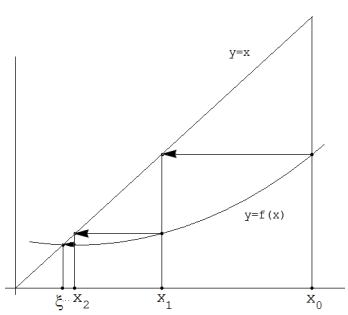


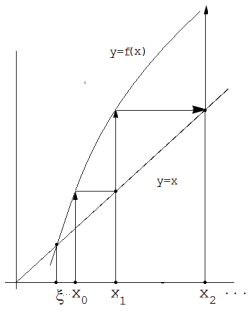
Доказательство. Покажем сначала, что $|x_n - \xi| < \frac{b-a}{3}$ для всех n. Применим метод математической индукции. При $n = 0$ справедливо неравенство $|x_0 - \xi| < \frac{b-a}{3}$ так как по условию теоремы x_0 и ξ принадлежат вместе отрезку $[\alpha, \beta]$ длины $\frac{b-a}{3}$. Пусть $|x_n - \xi| < \frac{b-a}{3}$. Тогда по формуле (3), формуле Лагранжа, условию теоремы и предположению индукции можно записать $|x_{n+1} - \xi| = |\varphi(x_n) - \varphi(\xi)| = |\varphi'(c)||x_n - \xi| < |x_n - \xi| < \frac{b-a}{3}$.

Так как каждый член последовательности удален от корня ξ меньше чем на треть отрезка $[a, b]$, а корень $\xi \in [\alpha, \beta]$, то все члены последовательности принадлежат отрезку $[a, b]$.

Следствие. Последовательность, удовлетворяющая условию теоремы 2 будет удовлетворять и условию первой теоремы 1, следовательно, для такой последовательности справедливы утверждения теоремы 1 и для ее членов справедлива оценка (6).

Геометрическая интерпретация условия сходимости. Условие (4) теоремы 1 нельзя отбросить. Поведение последовательности $\{x_n\}$ показано на следующих двух рисунках. Точка пересечения графиков $y = x$ и $y = \varphi(x)$ соответствует решению уравнения (2). На первом рисунке $0 < \varphi'(x) < 1$ и приведены первые три члена последовательности, по которым можно судить о сходимости последовательности к решению уравнения. На втором рисунке рассмотрено уравнение у которого $\varphi'(x) > 1$, последовательность $\{x_n\}$ удаляется от решения уравнения (2).





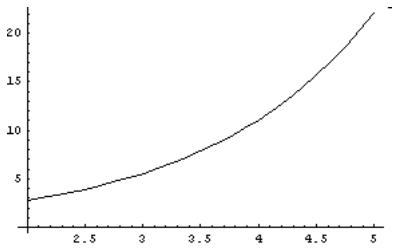
Уравнение (1) можно привести к виду $x = \varphi(x)$ несколькими способами, например, выразив x следующим образом: $x = 2^x - 10$. Обозначим правую часть

In[1]:= $\varphi[\mathbf{x\_}] = \mathbf{2^x - 10}$

Проверим условие сходимости итерационного процесса $|\varphi'(x)| \leqslant q < 1$ графически. Построим график функции $\varphi'[\mathbf{x}]$ на интервале изоляции корня.

In[2]:= $\mathbf{Plot}[\varphi'[\mathbf{x}], \{\mathbf{x}, \mathbf{2}, \mathbf{5}\}]$

Out[2]=



По графику видно, что модуль производной в окрестности нуля (≈ 4) уравнения (1) больше 1, следовательно условие сходимости не выполняется.

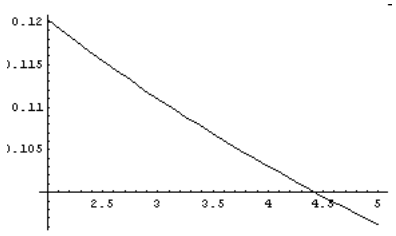
Выразим x из уравнения (1) другим способом: $x = Log[2, x + 10]$. Обозначим правую часть:

In[3]:= $\varphi[\mathbf{x\_}] = \mathbf{Log}[\mathbf{2}, \mathbf{x} + \mathbf{10}]$;

Проверим условие сходимости

In[4]:= $\mathbf{Plot}[\varphi'[\mathbf{x}], \{\mathbf{x}, \mathbf{2}, \mathbf{5}\}]$

Out[4]=



Смотрим на ось у-ов и видим, что условие сходимости $|\varphi'(\mathbf{x})| \leqslant \mathbf{q} < \mathbf{1}$ выполняется, например, с $\mathbf{q} = \frac{\mathbf{1}}{\mathbf{2}}$ на отрезке $[\mathbf{2}, \mathbf{5}]$.

Полагаем

In[5]:= $\mathbf{x} = \mathbf{b}$;

Запомним это значение:

In[6]:= $\mathbf{z} = \mathbf{x}$;

и построим второй член итерационной последовательности

In[7]:= $\mathbf{x} = \varphi[\mathbf{x}]//\mathbf{N};$

Проверим условие остановки вычислений:

In[8]:= $\mathbf{Abs}[\mathbf{x} - \mathbf{z}] < \mathbf{e}$

Если последняя команда вернет True, то найденное значение **x** есть корень уравнения с данной точностью, если False, то запоминаем второй элемент последовательности и генерируем третий, то есть повторяем три последние команды.

Если цикл повторяется многократно, то стоит объединить эти команды с помощью команды цикла *Do*. Команда ветвления If служит для завершения цикла в случае достижения нужной точности. При этом печатается элемент последовательности и шаг цикла, на котором достигнута данная точность. Команда $Break[]$ позволяет выйти из цикла в этом случае.

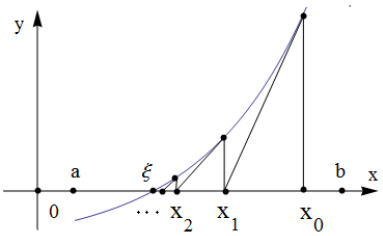
```
In[9]:= x = b;
        Do[
          z = x;
          x = φ[x]//N;
          If[Abs[z - x] < e,
           Print["Решение x = ", x//N," получено на ",i," шаге"];
           Break[]],
        {i, 1, 100}]
```

Out[9]= Решение x=3.78502 получено на 4 шаге.

Ответ: $x = 3.7850 \pm 0.0001$.

n.3 Метод Ньютона

Метод Ньютона (или метод касательных) является итерационным методом решения уравнения с одной неизвестной $f(x) = 0$ и характеризуется способом приведения этого уравнения к виду $x = \varphi(x)$. Метод Ньютона заключается в следующим.



Пусть функция $f(x)$, имеет единственный корень ξ на отрезке $[a, b]$. Возьмем точку $x_0 \in [a, b]$. Пусть касательная к графику функции $y = f(x)$ в точке с абсциссой x_0 пересекает ось x-ов в точке x_1. Пусть касательная к графику функции $y = f(x)$ в точке с абсциссой x_1 пересекает ось x-ов в точке x_2. Продолжая такие построения, получим последовательность точек $\{x_n\}_{n=0,1,2,..}$. При соблюдении некоторых условий (см. теорему) такая последовательность точек будет сходится к корню ξ.

Приведем формулу для вычисления элементов последовательности. Для этого напишем уравнение касательной к графику функции $y = f(x)$ в точке $(x_n, f(x_n))$:

$$y - f(x_n) = f'(x_n)(x - x_n).$$

Пересечение этой касательной с осью x-ов есть точка $(x_{n+1}, 0)$. Подставим координаты этой точки в уравнение касательной, получим $0 - f(x_n) = f'(x_n)(x_{n+1} - x_n)$. Отсюда находим

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}. \tag{7}$$

Теорема. Пусть $f(x)$ - дважды непрерывно дифференцируемая функция на отрезке $[a, b]$, произведение $f(a)f(b) < 0$ и производные $f'(x)$, $f''(x)$ сохраняют знак

на данном отрезке. Если точка x_0 такая, что $f(x_0)f''(x_0) > 0$, то последовательность $\{x_n\}_{n=0,1,2,\ldots}$, построенная по формуле (7) начиная с данного начального приближения x_0, сходится к корню уравнения $f(x) = 0$.

Доказательство. Из условия теоремы следует, что функция $f(x)$ строго монотонна и на концах отрезка принимает значения разных знаков. Поэтому уравнение $f(x) = 0$ имеет единственный корень на отрезке $[a, b]$. Обозначим этот корень через ξ.

Будем считать для определенности, что $f(a) < 0$, $f(b) > 0$, $f'(x) > 0$, $f''(x) > 0$ на отрезке $[a, b]$. Остальные выриванты знаков рассматриваются аналогично. Пусть x_0 такое как в условии теоремы. Тогда $f(x_0) > 0$.

Методом математической индукции докажем, что $x_n > \xi$ для всех n.

При $n = 0$ из неравенства $f(x_0) > 0$ и строго монотонного возрастания функции ($f'(x) > 0$ на отрезке $[a, b]$) следует, что $x_0 > \xi$.

Пусть $x_n > \xi$. Покажем, что $x_{n+1} > \xi$. Применяя формулу Тейлора с остаточным членом в форме Лагранжа, запишем, что

$$0 = f(\xi) = f(x_n + (\xi - x_n)) = f(x_n) + f'(x_n)(\xi - x_n) + \frac{1}{2}f''(c_n)(\xi - x_n)^2,$$

где $\xi \leqslant c_n \leqslant x_n$. В силе предположений теоремы и индукции последнее слагаемое в правой части этого равенства положительно, но тогда

$$f(x_n) + f'(x_n)(\xi - x_n) < 0 \quad \text{или} \quad -f(x_n) > f'(x_n)(\xi - x_n).$$

Отсюда и из (7) получим, что

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} > x_n + \xi - x_n = \xi.$$

Таким образом, доказали, что $x_n > \xi$ для всех n. Следовательно $f(x_n) > 0$ для всех n, а из (7) $x_{n+1} < x_n$. Значит последовательность $\{x_n\}_{n=1,2,\ldots}$, есть монотонно убывающая ограниченная снизу последовательность. Такая последовательность имеет предел, обозначим его $\overline{\xi} = \lim_{n\to\infty} x_n$. Перейдем, теперь, к пределу в (7) при $n \to \infty$, получим, что

$$\overline{\xi} = \overline{\xi} - \frac{f(\overline{\xi})}{f'(\overline{\xi})}$$

или $f(\overline{\xi}) = 0$. Функция $f(x)$ имеет единственный корень на отрезке $[a, b]$, поэтому $\overline{\xi} = \xi$ и теорема доказана.

Оценка погрешности $|\xi - x_n|$. Пусть задана предельная абсолютная погрешность $\varepsilon > 0$, с которой надо найти приближенное значение корня уравнения $f(x) = 0$, то есть требуется найти первый элемент итерационной последовательности $\{x_n\}$ такой, чтобы $|\xi - x_n| < \varepsilon$. Покажем как это сделать.

Так как $f(\xi) = 0$, то, применяя формулу Лагранжа, можно записать, что

$$|f(x_n)| = |f(\xi) - f(x_n)| = |f'(c)(\xi - x_n)| = |f'(c)||\xi - x_n| \geqslant m|\xi - x_n|,$$

где $m = \min_{x\in[a,b]} |f'(x)|$, $c \in [\xi, x_n]$. Отсюда получаем оценку

$$|\xi - x_n| \leqslant \frac{|f(x_n)|}{m}. \tag{8}$$

По формуле Тейлора

$$f(x_n) = f(x_{n-1} + (x_n - x_{n-1})) = f(x_{n-1}) + f'(x_{n-1})(x_n - x_{n-1}) + \frac{1}{2}f''(c)(x_n - x_{n-1})^2.$$

В силу (7) $f(x_{n-1}) + f'(x_{n-1})(x_n - x_{n-1}) = 0$, поэтому

$$|f(x_n)| = |\frac{1}{2}f''(c)(x_n - x_{n-1})^2| \leqslant M(x_n - x_{n-1})^2,$$

где $M = \max_{x\in[a,b]} |\frac{f''(x)}{2}|$. Отсюда и из (8) получаем оценку

$$|\xi - x_n| \leqslant \frac{M}{m}(x_n - x_{n-1})^2. \tag{9}$$

Так как при достаточно больших n справедливо неравенство

$$\frac{M}{m}(x_n - x_{n-1})^2 \leqslant |x_n - x_{n-1}|,$$

то из (9) окончательно получаем оценку погрешности в виде

$$|\xi - x_n| < |x_n - x_{n-1}|.$$

Оценка (9) показывает, что итерационная последовательность сходится к решению с квадратичной сходимостью.

Таким образом, для нахождения корня уравнения $f(x) = 0$ необходимо вычислять члены последовательности x_n по формуле (7), начиная с начального приближения x_0, до тех пор, пока модуль разности двух соседних членов последовательности не будет меньше наперед заданного ε : $|x_n - x_{n-1}| < \varepsilon$. Первый элемент последовательности x_n, удовлетворяющий такому условию, и будет являться приближенным значением корня уравнения $f(x) = 0$ с данной точностью ε.

Пример (Герон). Найти $x = \sqrt{a}$ с точностью $\varepsilon > 0$. Другими словами, требуется найти корень уравнения $x^2 - a = 0$ с данной точностью. Запишем формулу (7) для такого уравнения:

$$x_{n+1} = \frac{1}{2}(x_n + \frac{a}{x_n}). \tag{4}$$

Возьмем $a = 1.69$, $\varepsilon = 0.0001$. Корень будет принадлежать отрезку $[1, 1.5]$. Запишем (4):

$$x_{n+1} = \frac{1}{2}(x_n + \frac{1.69}{x_n}).$$

Пусть $x_0 = 1.5$. Находим $x_1 = \frac{1}{2}(1.5 + \frac{1.69}{1.5}) = 1.3133$. Проверим условие остановки итерационного процесса: $|x_1 - x_0| = |1.3133 - 1.5| = 0.1867 > 0.0001$. Поэтому находим $x_2 = \frac{1}{2}(1.3133 + \frac{1.69}{1.3133}) = 1.3007$. Проверим условие: $|x_2 - x_1| = |1.30007 - 1.3133| = 0.0133 > 0.0001$. Находим $x_3 = \frac{1}{2}(1.3001 + \frac{1.69}{1.3001}) = 1.3000$. Проверим условие: $|x_3 - x_2| = |1.30007 - 1.3000| = 0.000068 < 0.0001$. Таким образом, $x_3 = 1.3000$ есть приближенное значения корня, найденного с данной погрешностью.

Задача. В круглом колодце уровень воды от дна ровно один локоть. В него также опущены две тростинки: одна длиной 2 локтя, другая - 3 локтя. Тростинки

упираются нижним концом в края дна, верхним - в стену на противоположной стороне. Тростинки пересекаются ровно на уровне воды. Требуется определить диаметр колодца.

Это задача была на вступительном экзаменом для желающих стать жрецом в древнем Египте.

Решим задачу, сформулированную в начале параграфа, методом Ньютона. Первые четыре команды в начале параграфа считаем уже выполненными.

Проверим условие теоремы. Функция $f(x)$ на концах отрезка $[a,b]$ принимает значения разных знаков. Производные функции $f'[x] = 2^x log(2) - 1$ и $f''[x] = 2^x log^2(2)$ положительны на $[a,b]$. В качестве начальной точки итерационной последовательности можно взять конец отрезка b, так как выполняется условие теоремы

In[5]:= $\mathbf{f[b]f''[b] > 0}$

Out[5]= True

Полагаем

In[6]:= $\mathbf{x = b;}$

Запомним это значение

In[7]:= $\mathbf{z = x;}$

и построим второй член итерационной последовательности по формуле (7):

In[8]:= $\mathbf{x = (x - f[x]/f'[x])//N;}$

Проверим условие остановки вычислений

In[9]:= $\mathbf{Abs[x - z] < e}$

Если результат сравнения есть True, то найденное значение **x** есть корень уравнения с данной точностью, если False, то запоминаем второй элемент последовательности и генерируем третий, то есть повторяем три последние команды и так далее.

Если цикл повторяется многократно, то стоит объединить эти команды с помощью команды цикла *Do*. Перед циклом нужно выполнить команды 1-5.

```
In[6]:= x = b;
        Do[
          z = x;
          x = (x - f[x]/f'[x])//N;
          If[Abs[z - x] < e, Print["Решение  x = ", x//N,
           "  получено на  ", i, "  шаге."];
           Break[]],
        {i, 1, 100}]
```

Out[6]= Решение x=3.78503 получено на 4 шаге.

Решение следует округлить в соответствии с заданной точностью: $x = 3.7850 \pm 0.0001$.

Создадим команду $nut[f,x,b,e]$ возвращающую только ответ данной задачи. В качестве аргумента f в этой команде выступает левая часть уравнения (1), второй аргумент команды показывает как обозначен аргумент функции f, b - начальное приближение, e - точность.

Команда *Block* выделяет часть рабочего поля, в котором локальными переменными будут z,g. Цикл формируется командой While. Локальной переменной z присваивается первоначально значение $2e + b$. Тогда в команде $While$ условие $Abs[z - x] > e$ примет значение True первый раз и команда сработает как нужно. Если z ничего

не будет присвоено, то команда $While$ не сработает и команда $nut[f, x, b, e]$ вернет неверный ответ $x = b$.

Эта программа, как и все остальные, написана в стиле классического программирования. Аналогичную программу, написанную в стиле системы Mathematica, можно найти среди примеров в Help'е к команде FixedPoint.

```
In[1]:= nut[f_, x_, b_, e_] :=
        Block[{z = 2e + b, g},
          g = D[f, x];
          x = b;
          While[Abs[z - x] > e,
            z = x;
            x = x - f/g//N];
          x]
```

Перед применением команды нужно очистить переменную x от возможного предыдущего значения:

```
In[2]:= Clear[x]; nut[2^x - x - 10, x, 5, 0.001]
Out[2]= 3.78503
```

n.4 Метод деления отрезка пополам (метод бисекций)

Метод деления отрезка пополам позволяет найти корень уравнения $f(x) = 0$ на отрезке $[a, b]$ при условии, что функция $f(x)$ непрерывна и на концах отрезка принимает значения разных знаков: $f(a)f(b) < 0$.

Суть метода деления отрезка пополам заключена в следующем. Находим середину c отрезка $[a, b]$, содержащего корень данного уравнения, и проверяем условие $f(a)f(c) < 0$. Если условие есть True, то корень уравнения принадлежит отрезку $[a, c]$, который и принимаем за новый отрезок, полагая $b = c$, а если False, то корень уравнения принадлежит отрезку $[c, b]$, поэтому полагаем $a = c$. Проверяем условие остановки $|b - a| < \varepsilon$. Если оно верно, то c и есть корень с данной погрешностью. Если оценка не верна, то повторяем процесс деления пополам для нового отрезка $[a, b]$. Вариант $f(c) = 0$ сразу определяет корень.

Решаем уравнение (1). Вводим функцию, концы отрезка и точность

```
In[1]:= f[x_] := 2^x - x - 10;
In[2]:= a = 2;
        b = 5;
In[3]:= e = 0.001;
```

Находим середину отрезка:

```
In[4]:= c = (a + b)/2;
```

Функция $f[x]$ непрерывна на данном отрезке. Проверяем условие $f[a]f[c] < 0$ и выбираем половину отрезка

```
In[5]:= If[f[a] * f[c] < 0, b = c, If[f[c] == 0, Print["Решение x = ", c], a = c]]
```

Проверяем оценку длины отрезка

```
In[6]:= Abs[b - a] < e
```

Если условие не выполняется, то переходим на команду 4 и повторяем команды 4, 5, 6 до тех пор, пока условие 6 не будет True.

Для циклического выполнения этих команд можно применить команду Do, при этом, несколько изменим команду ветвления:

```
In[7]:= a = 2; b = 5;
In[8]:= Do[c = (a + b)/2.;
          fc = f[c];
          If[f[a] * fc < 0, b = c, If[fc ≠ 0, a = c]];
          If[Abs[b - a] < e||fc == 0,
          Print["Решение x = ", c//N, " получено на", i, "шаге."];
          Break[]],
         {i, 1, 100}]
Out[8]= Решение x=3.78491 получено на 12 шаге.
```

Следующая команда $bis[f, \{a1, b1\}, \varepsilon]$ находит корень функции f (обязательно с аргументом x) методом деления отрезка пополам, при этом, $\{a1, b1\}$ -список концов отрезка, а ε - точность определения корня. В организации цикла использована команда $While$, команда Block отделяет переменные команды bis от остальных переменных рабочего поля. Так, переменные a и b объявлены как локальные. Подстановка вида $f/.x \to a$ позволяет найти значение выражения (функции f) при $x = a$. Функция bis возвращает корень c.

```
In[1]:= bis[f_, {a1_, b1_}, ε_] :=
        Block[{a = a1, b = b1},
          While[Abs[b - a] > ε,
                c = (a + b)/2.;
                If[(f/.x → a)(f/.x → c) < 0, b = c,
                If[(f/.x → c) == 0, Break[], a = c]]
                ];
           c]
```

Применение функции:

```
In[2]:= Clear[x]; bis[2^x - x - 10, {2, 5}, 0.0001]
Out[2]= 3.7850
```

§14 Численное решение алгебраических уравнений

В этом параграфе определяются границы корней многочлена, рассматриваются методы Лобачевского-Грефе ([5]) и матричный метод определения корней многочлена с действительными коэффициентами.

n.1 Сведение кратных корней

Определение корней многочлена с кратными корнями можно свести к определению корней многочлена более низкой степени с различными корнями.

Рассмотрим многочлен

```
In[1]:= n = 5;  P[x_] = a0 ∏_{i=1}^{n} (x - z_i)^{α_i}
```

порядка $m = \alpha_1 + \alpha_2 + ... + \alpha_5$ с кратными корнями $z_1, ..., z_5$.

```
Out[1]= a0 (x - z1)^α1 (x - z2)^α2 (x - z3)^α3 (x - z4)^α4 (x - z5)^α5
```

Производная многочлена $P(x)$ равна
$P'(x) = a_0 (x - z_1)^{\alpha_1 - 1} (x - z_2)^{\alpha_2 - 1} (x - z_3)^{\alpha_3 - 1} (x - z_4)^{\alpha_4 - 1} (x - z_5)^{\alpha_5 - 1} Q(x)$,
где многочлен $Q(x)$ такой, что $Q(x_k) \neq 0$, $k = 1, 2, ..., 5$.

Поэтому многочлен
$R(x) = a_0 (x - z_1)^{\alpha_1 - 1} (x - z_2)^{\alpha_2 - 1} (x - z_3)^{\alpha_3 - 1} (x - z_4)^{\alpha_4 - 1} (x - z_5)^{\alpha_5 - 1}$
является наибольшим общим делителем многочленов $P(x)$ и $P'(x)$.

Такой многочлен можно найти с помощью алгоритма Евклида или воспользоваться встроенной командой:

In[2]:= $\mathbf{R[x\_] = PolynomialGCD[P[x], P'[x]]}$

Составим многочлен

In[3]:= $\mathbf{f[x\_] = Simplify[P[x]/R[x]]}$

Out[3]= $(x - z_1)(x - z_2)(x - z_3)(x - z_4)(x - z_5)$

Многочлен $f(x)$ имеет такие же корни, что и многочлен $P(x)$. Корни многочлена $f(x)$ не кратные.

Приведем пример. Пусть

In[1]:= $\mathbf{P[x\_] = x^8 + 29x^7 + 94x^6 - 4124x^5 - 32200x^4 + 166000x^3 + 1860000x^2 - 1800000x - 32400000;}$

Найдем нули многочлена.

In[2]:= $\mathbf{Solve[P[x] == 0, x]}$

Out[2]= $\{\{x \to -10\}, \{x \to -10\}, \{x \to -10\}, \{x \to -10\}, \{x \to -10\}, \{x \to 6\}, \{x \to 6\}, \{x \to 9\}\}$

Находим многочлены $R(x)$ и $f(x)$:

In[3]:= $\mathbf{R[x\_] = PolynomialGCD[P[x], P'[x]];}$
$\mathbf{f[x\_] = P[x]/R[x]//Simplify}$

Out[3]= $x^3 - 5x^2 - 96x + 540$

Для проверки найдем нули многочлена $f(x)$:

In[4]:= $\mathbf{Solve[f[x] == 0, x]}$

Out[4]= $\{\{x \to -10\}, \{x \to 6\}, \{x \to 9\}\}$

п.2 Границы корней

Дадим грубую оценку модулей корней многочлена с действительными коэффициентами

$$P(x) = a_0 x^n + a_1 x^{n-1} + ... + a_{n-1} x + a_n \tag{1}$$

Пусть $A = \max\{|a_1|, |a_2|, ..., |a_n|\}$. Тогда модули всех корней x_k, $k = 1, ..., n$, многочлена (1) удовлетворяют неравенству

$$|x_k| < 1 + \frac{A}{|a_0|}. \tag{2}$$

Пусть $a_n \neq 0$, $B = \max\{|a_0|, |a_1|, ..., |a_{n-1}|\}$. Тогда модули всех корней x_k, $k = 1, ..., n$, многочлена (1) удовлетворяют неравенству

$$|x_k| > \frac{1}{1 + \frac{B}{|a_n|}}. \tag{3}$$

Рассмотрим пример. Пусть дан многочлен.

In[1]:= $\mathbf{P[x\_] = 2x^5 - 100x^2 + 2x - 1;}$

Составим список коэффициентов многочлена

In[2]:= $\mathbf{q = CoefficientList[P[x], x]}$

Out[2]= {-1, 2, -100, 0, 0, 2}

Найдем максимальные значения коэффициентов.

In[3]:= $\mathbf{A = Max[Abs[Drop[q, -1]]]}$
$\mathbf{B = Max[Abs[Drop[q, 1]]];}$

Out[3]= 100
100

По формулам (2) и (3) найдем концы отрезка, содержащего корни многочлена (1):

In[4]:= $\mathbf{gr = \{\frac{1}{1 + \frac{B}{Abs[First[q]]}}, 1 + \frac{A}{Abs[Last[q]]}\}//N}$

Out[4]= {0.00990099,51.}

Следующий "принцип аргумента" позволяет узнать число корней многочлена, расположенных внутри замкнутого контура на комплексной плоскости.

Если многочлен $P(x)$ не имеет корней на замкнутом контуре γ, то число корней многочлен $P(x)$, лежащих внутри контура γ, равно изменению аргумента $\arg P(x)$, при положительном обходе контура γ, деленному на 2π. При этом каждый корень считается столько раз, какова его кратность.

Если уравнение контура γ есть

$$x = \xi(t) + i\eta(t), \quad t \in [0, t_0],$$

то изменение аргумента $\arg P(x)$, при положительном обходе контура γ, деленному на 2π, будет равно числу оборотов, которых делает кривая

$$P(\xi(t) + i\eta(t)) = x(t) + i\,y(t), \quad t \in [0, t_0] \tag{4}$$

вокруг начала координат.

Рассмотрим пример. Пусть дан многочлен.

In[1]:= $\mathbf{P[x\_] = x^3 - 3x + 1;}$

Найдем число его корней, принадлежащих кругу радиуса 2 с центром в начале координат.

Так как уравнение окружности радиуса 2 с центром в начале координат на комплексной плоскости $x = 2(\cos(t) + i sin(t))$, $t \in [0, 2\pi]$, то уравнение кривой (4) примет вид

In[2]:= $\mathbf{p[t\_] = P[2(Cos[t] + I\,Sin[t])]//Expand}$

Out[2]= $8cos^3(t) + 24isin(t)cos^2(t) - 24sin^2(t)cos(t) - 6\,cos(t) - 8isin^3(t) - 6isin(t) + 1$

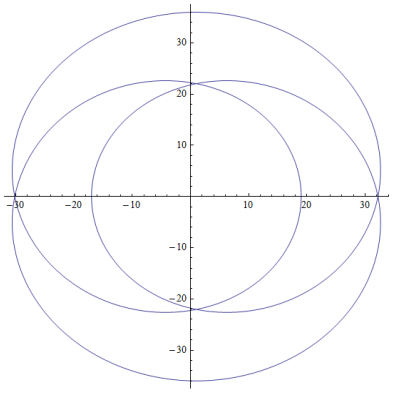
Выделим действительную и мнимую части и напишем параметрические уравнения кривой (1):

In[3]:= $\mathbf{x[t\_] = Simplify[Re[p[t]], Element[t, Reals]]}$
$\mathbf{y[t\_] = Simplify[Im[p[t]], Element[t, Reals]]}$

Out[3]= $-6cos(t) + 8cos(3t) + 1$
$2(8cos(2t) + 1)sin(t)$

Теперь можно построить кривую

In[4]:= $\mathbf{ParametricPlot[\{x[t], y[t]\}, \{t, -Pi, Pi\}, AspectRatio -> 1]}$



Видим, что число оборотов кривой около начала координат равно 3. Следовательно данный многочлен имеет три корня в круге радиуса 2.

Для сравнения, найдем нули данного многочлена:

In[5]:= $\mathbf{Solve[P[x] == 0, x]//N//Chop}$

Out[5]= $\{\{x- > 1.53209\}, \{x- > 0.347296\}, \{x- > -1.87939\}\}$

Приведем еще два простых аналитических способа определения числа корней многочлена, лежащих внутри замкнутого контура, в рассмотренном примере.

Найдем производную аргумента кривой (4):

In[6]:= $\mathbf{d[t\_] = D[ArcTan[y[t]/x[t]], t]//FullSimplify}$

Out[6]= $(6(cos(t) + 32cos(2t) - 4cos(3t) - 38))/(12cos(t) + 96cos(2t) - 16\ cos(3t) - 101)$

Интеграл от этой производной от $[0, 2\pi]$, деленный на 2π, даст нам число оборотов кривой (4) около начала координат:

In[7]:= $\mathbf{NIntegrate[d[t], \{t, 0, 2Pi\}]/(2Pi)}$

Out[7]= 3

Еще один способ. Введем длину вектора

In[8]:= $\mathbf{nr[x\_] = Sqrt[x.x]}$

и рассмотрим последовательность точек на кривой (4), идущих с шагом, например, 0.01:

In[8]:= $\mathbf{q = Table[\{x[t], y[t]\}, \{t, 0, 2Pi, 0.01\}];}$

Просуммируем все углы с центром в начале координат, стороны которых проходят через две соседние точки последовательность $q[[i]]$ и $q[[i+1]]$. Угол находим при помощи формулы скалярного произведения векторов. При этом учитываем знак угла - если базис $\{q[[i]], q[[i+1]]\}$ положительный, то угол положительный и наоборот. Полученную сумму делим на 2π и округляем до целых чисел.

In[9]:=

$$\mathbf{Round}\left[\frac{\mathbf{1}}{\mathbf{2Pi}} \sum_{\mathbf{i=1}}^{\mathbf{Length[q]-1}} \mathbf{Sign[Det[\{q[[i]], q[[i+1]]\}]]ArcCos}\left[\frac{\mathbf{q[[i]].q[[i+1]]}}{\mathbf{nr[q[[i]]]nr[q[[i+1]]]}}\right]\right]$$

Out[9]= 3

И в этом случае число корней многочлена примера в круге радиуса 2 с центром в начале координат равно 3.

n.3 Метод знакопеременных сумм

Данный метод позволяет найти границы положительных корней многочлена.

Пусть у многочлена

$$P(x) = a_0x^n + a_1x^{n-1} + ... + a_{n-1}x + a_n \tag{1}$$

старший коэффициент $a_0 > 0$. Представим многочлен в виде знакопеременной суммы. Пусть $Q_1(x)-$ сумма последовательных членов многочлена $P(x)$ с положительными коэффициентами, начиная с a_0x^n, $-Q_2(x)-$ сумма следующих за $Q_1(x)$ последовательных членов многочлена $P(x)$ с отрицательными коэффициентами и так далее, в результате получим следующее представление

$$P(x) = Q_1(x) - Q_2(x) + Q_3(x) - Q_4(x) + ... + Q_{2m-1} - Q_{2m}, \tag{2}$$

где последнее слагаемое $-Q_{2m}(x)$ или состоит из членов с отрицательными коэффициентами или тождественно равен нулю.

Пусть c_j, $j = 1, 2, ..., m$, положительные числа такие, что

$$Q_{2j-1}(c_j) - Q_{2j}(c_j) \geqslant 0.$$

Тогда за верхнюю границу положительных нулей многочлена $P(x)$ можно принять число

$$R = \max(c_1, c_2, ..., c_m).$$

Для нахождения нижней границы положительных корней многочлена $P(x)$ сделаем замену переменной в многочлене $P(x)$ по правилу $x \to \frac{1}{x}$. Если R_1 есть верхняя граница положительных корней уравнения

$$P(\frac{1}{x}) = 0,$$

то число

$$r = \frac{1}{R_1}$$

будет нижней границей положительных корней многочлена $P(x)$.

Рассмотрим пример. Для многочлена

In[1]:= $\mathbf{P[x\_] = 2x^5 - 100x^2 + 2x - 1}$

найдем границы для действительных и положительных корней.

Составим список коэффициентов данного многочлена

In[2]:= $\mathbf{p = Reverse[CoefficientList[P[x], x]]}$

Out[2]= {2,0,0,-100,2,-1}

Проверим знак старшего коэффициента многочлена и, если он отрицателен, то сменим знаки коэффициентов на противоположный.

In[3]:= $\mathbf{If[First[p] < 0, p = p\,Sign[First[p]]]}$

Для завершения следующего цикла добавим к списку коэффициентов последний коэффициент с противоположным знаком:

In[4]:= $\mathbf{p = Append[p, -Last[p]]}$

Формируем многочлены Q_k :

In[5]:= $\mathbf{k = 1; m = 1; n = Length[p] - 1;}$

In[6]:= $\mathbf{Do[If[Sign[p[[i]]] == (-1)^k, Q_k[x\_] = (-1)^{k+1} \sum_{j=m}^{i-1} p[[j]] x^{n-j}; k = k+1; m = i;], \{i, 1, n+1\}];}$

Добавим последний многочлен нулевой

In[7]:= $\mathbf{If[Last[p] < 0, Q_k[x\_] = 0, k = k-1];}$

Сделаем проверку, найдем разность

In[8]:= $\mathbf{Sum[(-1)^{i+1} Q_i[x], \{i, 1, k\}] - P[x]}$

Out[8]= 0

Находим положительные числа, на которых разности положительны.

In[9]:= $\mathbf{mn = Table[Minimize[\{Q_{2i-1}[x] - Q_{2i}[x], Q_{2i-1}[x] - Q_{2i}[x] >= 0 \&\& x > 0\}, x]//N, \{i, 1, k/2\}]}$

Out[9]= $\{\{2.27374 \times 10^{-13}, \{x \to 3.68403\}\}, \{0., \{x \to 0.5\}\}\}$

Выбираем наибольшее значения найденных x-ов и получаем верхнюю границу положительных корней многочлена:

In[10]:= $\mathbf{Max[Table[mn[[i, 2, 1, 2]], \{i, 1, Length[mn]\}]]}$

Out[10]= 3.68403

Для сравнения найдем корни многочлена встроенной командой

In[11]:= $\mathbf{Solve[P[x] == 0, x]//N//Chop}$

Out[11]= $\{\{x \to 3.67825\}, \{x \to -1.84913 - 3.18971i\}, \{x \to -1.84913 + 3.18971i\}, \{x \to 0.0100009 - 0.0994983i\}, \{x \to 0.0100009 + 0.0994983i\}\}$

Найдем нижнюю границу положительных корней. Для этого сделаем в многочлене $P[x]$ замену $x \to 1/x$:

In[1]:= $\mathbf{P[x\_] = P[\frac{1}{x}] x^5//FullSimplify//Expand}$

Out[1]= $2 - 100x^3 + 2x^4 - x^5$

Найдем верхнюю границу положительных корней этого многочлена, то есть выполним команды 2-9:

In[2]:= $\mathbf{p = Reverse[CoefficientList[P[x], x]];}$

In[3]:= $\mathbf{If[First[p] < 0, p = p\ Sign[First[p]]];}$

In[4]:= $\mathbf{p = Append[p, -Last[p]];}$

In[5]:= $\mathbf{k = 1; m = 1; n = Length[p] - 1;}$

In[6]:= $\mathbf{Do[If[Sign[p[[i]]] == (-1)^k, Q_k[x\_] = (-1)^{k+1} \sum_{j=m}^{i-1} p[[j]] x^{n-j}; k = k+1; m = i;], \{i, 1, n+1\}];}$

In[7]:= $\mathbf{If[Last[p] < 0, Q_k[x\_] = 0, k = k-1];}$

In[8]:= $\mathbf{mn = Table[Minimize[\{Q_{2i-1}[x] - Q_{2i}[x], Q_{2i-1}[x] - Q_{2i}[x] >= 0 \&\& x > 0\}, x]//N, \{i, 1, k/2\}]}$

In[9]:= $\mathbf{Max[Table[mn[[i, 2, 1, 2]], \{i, 1, Length[mn]\}]]}$

Out[9]= 2

Тогда нижняя граница положительных корней будет равна

In[10]:= $\mathbf{1/\%}$

Out[10]= 0.5

Таким образом, положительные корни многочлена $P[x]$ принадлежат отрезку $[0.5, 3.68403]$.

п.4 Число действительных корней. Теорема Штурма

Для многочлена $P(x)$ обозначим через $P_1(x) = P'(x)$, $P_2(x)-$ взятый с обратным знаком остаток при делении многочлена $P(x)$ на $P_1(x)$, $P_3(x)-$ взятый с обратным

знаком остаток при делении многочлена $P_1(x)$ на $P_2(x)$ и так далее. В результате получим последовательность многочленов Штурма

$$P(x),\ P_1(x),\ P_2(x), ..., P_m(x). \tag{1}$$

Обозначим через N_c число перемен знака в системе Штурма (1) при $x = c$, то есть число смены знаков у соседних элементов числовой последовательности. При этом, нулевые элементы не рассматриваются.

Теорема Штурма. Если многочлен $P(x)$ порядка n не имеет кратных корней и $P(a) \neq 0$, $P(b) \neq 0$, то число его действительных корней $N_{a,b}$ на отрезке $a < x < b$ равно числу перемен знака в системе Штурма многочлена $P(x)$ при переходе от $x = a$ к $x = b$, то есть

$$N_{a,b} = N_a - N_b. \tag{2}$$

В частности,

1) если $P(0) \neq 0$, то число N_+ положительных и число N_- отрицательных корней многочлена $P(x)$ соответственно равны

$$N_+ = N_0 - N_\infty, \quad N_- = N_{-\infty} - N_0.$$

2) все корни многочлена $P(x)$ действительны, если $N_{-\infty} - N_\infty = n$.

С помощью теоремы Штурма можно отделять корни алгебраических уравнений, разбивая интервал, содержащий все действительные корни уравнения на конечное число интервалов (α, β) таких, что $N_\alpha - N_\beta = 1$.

Рассмотрим пример. Определить число положительных и отрицательных корней многочлена

In[1]:= $\mathbf{P[x\_] = x^4 - 4x + 1;}$

Перейдем к многочлену с различными корнями (п. 1).

In[2]:= $\mathbf{R[x\_] = PolynomialGCD[P[x], P'[x]];}$
$\mathbf{P[x\_] = P[x]/R[x]//Simplify}$

Out[2]= $x^4 - 4x + 1;$

Видим, что исходный многочлен не имеет кратных корней.

Введем первые два многочлена последовательности Штурма.

In[2]:= $\mathbf{P_0[x\_] = P[x];}$
$\mathbf{P_1[x\_] = P'[x];}$

Остальные многочлены последовательности Штурма получим из следующего цикла. Остаток при делении многочленов дает команда PolynomialRemainder.

In[3]:=
```
i = 0; g = P0[x];
While[Not[NumberQ[g]],
  Pi+2[x_] = -PolynomialRemainder[Pi[x], Pi+1[x], x]//N;
  g = Pi+2[x];
  i++]
```

Теперь можно составить систему многочленов Штурма

In[4]:= $\mathbf{f[x\_] = Table[P_k[x], \{k, 0, i + 1\}]}$

Out[4]= $\{x^4 - 4x + 1, 4x^3 - 4, 3.x - 1., 3.85185\}$

Для подсчета числа перемен знаков в списке v составим команду

In[5]:= $\mathbf{zn[v\_] := Block[\{j = 0, u\}, u = DeleteCases[v, 0];}$

$$\mathbf{Do[If[u[[i]]u[[i+1]]<0,j++],\{i,1,Length[u]-1\}];j]}$$

Здесь команда $u = DeleteCases[v, 0]$ убирает нули из списка v, затем в цикле вычисляется число перемен знака.

Найдем значения многочленов Штурма на бесконечности и в точке 0:

```
In[6]:= q_{-∞} = Sign[Limit[f[x], x-> -∞]]
        q_{+∞} = Sign[Limit[f[x], x-> ∞]]
        q_0 = Sign[f[0]]
Out[6]= {1,-1,-1,1}
        {1,1,1,1}
        {1,-1,-1,1}
```

Подсчитаем число перемен знаков в данных списках.

```
In[7]:= N_{-∞} = zn[q_{-∞}]
        N_{+∞} = zn[q_{+∞}]
        N_0 = zn[q_0]
Out[7]= 2
        0
        2
```

Определяем число положительных и отрицательных корней исходного многочлена.

```
In[8]:= N_+ = N_0 - N_{+∞}
        N_- = N_{-∞} - N_0
Out[8]= 2
        0
```

Таким образом, многочлен имеет 2 действительных положительных корня, значит, остальные 2 корня - комплексные.

n.5 Теорема Бюдана-Фурье

Система Штурма требует определенных вычислений, поэтому на практике применяют более простую схему подсчета действительных корней многочлена.

Пусть $\underline{N}$ есть число перемен знака в числовой последовательности q, не содержащей нулевых элементов. Это число назовем нижним числом перемен знака.

Пусть в последовательности q есть нулевые элементы, например $q_k = q_{k+1} = ...q_{k+l-1}$, но $q_{k-1} \neq 0$, $q_{k+l} \neq 0$. Заменим эти нулевые элементы на элементы $\widetilde{q}_{k+i} = \pm 1$, $i = 0, 1, ..., l-1$, такие, что

$$\operatorname{sign}(\widetilde{q}_{k+i}) = (-1)^{l-i} \operatorname{sign}(q_{k+l}). \tag{1}$$

Предполагаем, что первый и последний элементы последовательности q не нулевые.

Число $\overline{N}$ перемен знака в такой новой последовательности назовем верхним числом перемен знака последовательности q.

Если в последовательности q нет нулевых элементов, то $\underline{N} = \overline{N}$.

Теорема Бюдана-Фурье. Если числа a и b $(a < b)$ не являются корнями многочлена $P(x)$ степени n, то число $N_{a,b}$ действительных корней уравнения $P(x) = 0$ расположенных между a и b, равно минимальному числу $\triangle N$ перемен знаков, в системе последовательных производных

$$P(x),\ P'(x),\ P^{(n-1)}(x), ..., P^{(n)}(x) \tag{2}$$

при переходе от $x = a$ к $x = b$, или меньше числа $\triangle N$ на четное число, то есть

$$N_{a,b} = \triangle N - 2k,$$

где

$$\triangle N = \underline{N}_a - \overline{N}_b$$

и $\underline{N}_a$– нижнее число перемен знаков в системе (1) при $x = a$, $\overline{N}_b$– верхнее число перемен знаков в этой системе при $x = b$.

Теорема Декарта. Число положительных корней многочлена

$$a_0x^n + a_1x^{n-1} + ... + a_{n-1}x + a_n, \quad (a_0 \neq 0) \tag{3}$$

с учетом их кратности равно числу перемен знаков в системе коэффициентов a_0, $a_1, ..., a_n$ (коэффициенты, равные нулю не рассматриваются) или меньше этого числа на четное число.

Теорема Гюа. Если уравнение

$$a_0x^n + a_1x^{n-1} + ... + a_{n-1}x + a_n = 0$$

имеет действительные коэффициенты и все корни его действительны, то квадрат каждого некрайнего коэффициента этого уравнения больше произведения двух его соседних коэффициентов, то есть выполняется неравенство:

$$a_k^2 > a_{k-1}a_{k+1}, \quad k = 1, 2, ..., n-1. \tag{4}$$

Если это неравенство нарушается при некотором k, то многочлен имеет по крайней мере одну пару комплексных корней.

Введем команды подсчета числа перемен знака. Сначала введем команду преобразования последовательности в соответствии с правилом (1).

Рассмотрим для примера последовательность.

In[1]:=
```
v = {1, 0, 0, -3, 0, 0, 0, 0, 4, 1, 0, 4, 5, 0, 0, 0, 0, 4, 4}
```

Заменим в этой последовательности нули по правилу (1):

In[2]:=
```
u = v; j = 0;
```

In[3]:=
```
Do[
    If[u[[i]] == 0,
    j++,
    If[j > 0,
    Do[u[[k]] = (-1)^(j-k) Sign[u[[i]]], {k, i - j, i - 1}];
    j = 0]],
    {i, 2, Length[u] - 1}];
```

В результате получим преобразованную последовательность:

In[4]:=
```
u
```

Out[4]=
```
{1, -1, 1, -3, -1, 1, -1, 1, 4, 1, 1, 4, 5, 1, -1, 1, -1, 4, 4}
```

Теперь вводим команду для подсчета верхнего числа перемен знака. По заданной последовательности строится последовательность, в которой нули заменяются по правилу (1) и подсчитывается число перемен знака в новой последовательности.

In[5]:=
```
zv[v_] := Block[{j = 0, u = v},
    Do[
```

```
    If[u[[i]] == 0,
    j++,
    If[j > 0,
    Do[u[[k]] = (-1)^(j-k) Sign[u[[i]]], {k, i - j, i - 1}];
    j = 0]],
    {i, 2, Length[u] - 1}];
    Do[If[u[[i]]u[[i + 1]] < 0, j++], {i, 1, Length[u] - 1}]; j]
```

Вводим команду для подсчета нижнего числа перемен знака:

```
In[6]:= zn[v_] := Block[{j = 0, u}, u = DeleteCases[v, 0];
    Do[If[u[[i]]u[[i + 1]] < 0, j++], {i, 1, Length[u] - 1}]; j]
```

Введем команду вычисления степени многочлена:

```
In[7]:= deg[m_] := Block[{k = 1, xx = Variables[m][[1]]},
    While[Not[D[m, xx, k] === 0], k++]; k - 1]
```

Рассмотрим пример. Найдем число действительных корней многочлена

```
In[8]:= P[x_] = -20 + 104x - 12x^2 + 8x^3 + x^4;
```

Проверим наличие комплексных корней данного многочлена. Применим теорему Гуа. Составим список коэффициентов многочлена.

```
In[9]:= q = CoefficientList[P[x], x]
Out[9]= {-20,104,-12,8,1}
```

Проверим выполнение неравенств (4).

```
In[10]:= Table[q[[i]]^2 > q[[i - 1]]q[[i + 1]], {i, 2, Length[q] - 1}]
Out[10]= {True,False,True}
```

По теореме Гуа есть комплексные корни. Так как комплексных корней четное число, то действительных корней многочлена не больше двух.

Применим теорему Декарта и подсчитаем число положительных корней. Для этого подсчитаем число перемен знака в последовательности q коэффициентов многочлена.

```
In[11]:= zn[q]
Out[11]= 3
```

Число положительных корней либо равно 3, либо отличается от 3 на четное число. Следовательно число положительных корней равно 1.

Таким образом, данное уравнение имеет 2 комплексных корня, один положительный и один отрицательный корень.

Теорема Бюдана - Фурье позволяет узнать число действительных корней, например, на интервале (0, 2).

Найдем степень многочлена и составим последовательность производных (2):

```
In[12]:= Clear[m]; n = deg[P[x]];
In[13]:= m[x_] = Table[D[P[x], {x, k}], {k, 0, n}]
Out[13]= {-20 + 104x - 12x^2 + 8x^3 + x^4, 104 - 24x + 24x^2 + 4x^3,
    -24 + 48x + 12x^2, 48 + 24x, 24}
```

Найдем разность

```
In[14]:= zn[m[0]] - zv[m[2]]
Out[14]= 3
```

Число действительных корней на интервале (0,2) либо равно 3, либо отличается от 3 на четное число. Значит число корней на данном интервале равно 1.

n.6 Метод Лобачевского-Греффе

Идея Метода Лобачевского-Грефе. Пусть уравнение

$$a_0x^n + a_1x^{n-1} + ... + a_{n-1}x + a_n = 0 \tag{1}$$

имеет n различных действительных корней, модули которых значительно отделены друг от друга

$$|x_1| \gg |x_2| \gg |x_3| \gg ... \gg |x_n|, \tag{2}$$

то есть считаем, что отношение двух соседних корней

$$\varepsilon_i = \frac{x_{i+1}}{x_i} \tag{3}$$

оценивается по модулю малой величиной $|\varepsilon_i| < \varepsilon$, $i = 1, 2, ..., n-1$.

Из (1) по теореме Виета можно записать

$$\begin{aligned} x_1 + x_2 + ... + x_n &= -\tfrac{a_1}{a_0}, \\ x_1x_2 + x_1x_3 + ... + x_{n-1}x_n &= \tfrac{a_2}{a_0}, \\ . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \\ x_1x_2...x_n &= (-1)^n\tfrac{a_1}{a_0}. \end{aligned}$$

Отсюда в силу (3) можно записать

$$\begin{aligned} x_1(1 + e_1) &= -\tfrac{a_1}{a_0}, \\ x_1x_2(1 + e_2) &= \tfrac{a_2}{a_0}, \\ . \quad . \quad . \quad . \quad . \\ x_1x_2...x_n &= (-1)^n\tfrac{a_1}{a_0}, \end{aligned}$$

где e_i достаточно малые величины. Пренебрегая в этих равенствах малыми величинами e_i, получим следующие равенства

$$\begin{aligned} x_1 &= -\tfrac{a_1}{a_0}, \\ x_1x_2 &= \tfrac{a_2}{a_0}, \\ . \quad . \quad . \quad . \quad . \\ x_1x_2...x_n &= (-1)^n\tfrac{a_1}{a_0}, \end{aligned}$$

откуда получаем искомые корни приближенно равными

$$\begin{aligned} x_1 &= -\tfrac{a_1}{a_0}, \\ x_2 &= -\tfrac{a_2}{a_1}, \\ . \quad . \quad . \\ x_n &= -\tfrac{a_n}{a_{n-1}}. \end{aligned} \tag{4}$$

Идея метода заключается в составлении уравнения, корни которого являются степенями корней данного уравнения. Степень подбирается достаточно большой настолько, чтобы для нового уравнения выполнялось свойство отделимости корней (2). Определяем корни нового уравнения по формулам (4), извлекаем из этих корней соответствующие корни (случай действительных корней) и получаем корни данного уравнения.

Переход от данного уравнения к уравнению, корни которого отделены называется квадрированием корней.

Процесс квадрирования корней.

Покажем, как можно построить многочлен, корни которого есть квадраты корней данного многочлена, взятые со знаком минус.

Рассмотрим многочлен

$$P(x) = a_0x^n + a_1x^{n-1} + ... + a_{n-1}x + a_n.$$

Если x_1, x_2, ...,x_n - корни этого многочлена, то можно записать

$$P(x) = a_0(x - x_1)(x - x_2)...(x - x_n).$$

Так как

$$P(-x) = (-1)^n a_0(x + x_1)(x + x_2)...(x + x_n),$$

то

$$P(x)P(-x) = (-1)^n a_0^2(x^2 - x_1^2)(x^2 - x_2^2)...(x^2 - x_n^2). \tag{5}$$

Полагая $y = -x^2$ в (5) получим многочлен

$$Q(y) = a_0^2(y + x_1^2)(y + x_2^2)...(y + x_n^2). \tag{6}$$

с корнями $y_k = -x_k^2$, $k = 1, 2, ..., n$.

Найдем коэффициенты многочлена (6). Так как

$$P(-x) = (-1)^n[a_0x^n - a_1x^{n-1} + a_2x^{n-2} - ... + (-1)^n a_n],$$

то

$$P(x)P(-x) = (-1)^n[a_0^2x^{2n} - (a_1^2 - 2a_0a_2)x^{2n-2} + (a_2^2 - 2a_1a_3 + 2a_0a_4)x^{2n-4} - ... + (-1)^n a_n^2].$$

Приравниваем этот многочлен к нулю и получаем уравнение

$$A_0y^n + A_1y^{n-1} + ... + A_{n-1}y + A_n = 0, \tag{7}$$

где

$$A_k = a_k^2 + 2\sum_{s=1}^{k}(-1)^s a_{k-s}a_{k+s}, \quad k = 0, 1, 2, ..., n, \tag{8}$$

а

$$a_s = 0 \quad \text{при} \quad s > n. \tag{9}$$

Рассмотрим пример квадрирования корней. Рассмотрим многочлен и введем его степень.

In[1]:= $\mathbf{P[x\_] = x^3 - 3x + 1; n = 3;}$

Составим список коэффициентов многочлена.

In[2]:= $\mathbf{q = CoefficientList[P[x], x]}$

Out[2]= {1,-3,0,1}

Введем обозначение коэффициентов для формулы (8).

In[3]:= $\mathbf{Table[a_{n-s} = q[[s + 1]], \{s, 0, n\}]}$

Введем нулевые коэффициенты в соответствии с (9).

In[4]:= $\mathbf{Table[a_s = 0, \{s, n+1, 2n\}]}$;

Теперь можно подсчитать коэффициенты нового уравнения по формуле (8).

In[5]:= $\mathbf{Table[A_i = a_i{}^2 + 2\sum_{k=1}^{i}(-1)^k a_{i-k}a_{i+k}, \{i, 0, n\}]}$

Out[5]= {1,6,9,1}

Новый многочлен, корни которого есть квадраты корней многочлена $P[x]$ взятые со знаком минус.

In[6]:= $\mathbf{Q[x\_] = \sum_{i=0}^{n} A_{n-i}x^i}$

Out[6]= $1 + 9x + 6x^2 + x^3$

Для проверки найдем корни многочленов встроенной командой.

In[7]:= $\mathbf{m1 = Solve[P[x] == 0, x]//N//Chop}$
$\mathbf{Solve[Q[x] == 0, x]//N//Chop}$

Out[7]= $\{\{x \to 1.53209\}, \{x \to 0.347296\}, \{x \to -1.87939\}\}$
$\{\{x \to -0.120615\}, \{x \to -3.53209\}, \{x \to -2.3473\}\}$

Возведем корни многочлена $P[x]$ в квадрат и сравним с корнями квадрированного уравнения.

In[8]:= $\mathbf{Power[\#[[1, 2]], 2]\&/@m1//Chop}$

Out[8]= {2.3473,0.120615,3.53209}

Повторяя процесс для многочлена $Q[x]$, получим многочлен корни которого есть четвертые степени исходного многочлена. Будем повторять процесс квадрирования до тех пор, пока в формуле (8) сумма

$$2\sum_{s=1}^{k}(-1)^s a_{k-s}a_{k+s}$$

перестанет влиять на коэффициент a_k^2 или отношение

$$\frac{\sum_{k=1}^{i}(-1)^k a_{i-k}a_{i+k}}{a_i^2} \tag{10}$$

в формуле (8) становится достаточно малым.

Используем команду цикла и составим многочлен, корни которого есть $2^7 = 128$ степени корней исходного многочлена. Цикл проработает 7 раз. Первая команда Print выводит на экран отношения (10), вторая - список коэффициентов q квадрированного многочлена да данном шаге.

In[1]:= $\mathbf{P[x\_] = Q[x\_] = x^3 - 3x + 1; n = 3;}$

In[2]:=
```
n = 3;
Do[q = CoefficientList[P[x], x]
  Table[a_(n-s) = q[[s + 1]], {s, 0, n}]
  Table[a_s = 0, {s, n + 1, 2n}];
  Table[A_i = a_i^2 + 2 Sum_(k=1)^i (-1)^k a_(i-k) a_(i+k), {i, 0, n}]
  Print[Table[If[a_i ≠ 0, Sum_(k=1)^i (-1)^k a_(i-k) a_(i+k)/a_i^2, 1], {i, 0, n}]//N//Chop];
  P[x_] = Sum_(i=0)^n A_(n-i) x^i,
  Print["q = ", q = N[Reverse[CoefficientList[P[x], x]], 5]],
 {7}]
```

Out[2]= {0,2.,0,0}
u={1.0000,6,9.0000,1.0000}

{0,-0.25,-0.07407,0}
u={1.0000,18.000,69.000,1.0000}
{0,-0.213,-0.00378,0}
u={1.0000,186,4725.0,1.0000}
$\{0, -0.1366, -8.331234 \times 10^{-11}, 0\}$
$u = \{1.0000, 25146., 22325253, 1.0000\}$
$\{0, -0.03531, -5.045174\,10^{-11}, 0\}$
$u = \{1.0000, 587670810, 498416921463717, 1.0000\}$
$\{0, -0.001443, -2.365639 * 10^{-21}, 0\}$
$u = \{1, 344360147083128666, 248419427601369039924572114469, 1\}$
$\{0, -2.094883 * 10^{-}6, -5.580097 * 10^{-}42, 0\}$
$u = \{1.0000, 1.1858 \times 10^{35}, 6.1712 \times 10^{58}, 1.0000\}$

Отношения (10) стали достаточно малыми $\{0, -4.18977 \times 10^{-6}, 0, 0\}$ на 7 шаге цикла. Теперь по формуле (4) находим список корней последнего квадрированного многочлена и извлекаем из этого списка корень степени 128. Получим список корней исходного многочлена взятых по абсолютной величине.

In[3]:= $\mathbf{w} = (\mathbf{Table}[\mathbf{q}[[\mathbf{i}]]/\mathbf{q}[[\mathbf{i}-\mathbf{1}]], \{\mathbf{i}, \mathbf{2}, \mathbf{n}+\mathbf{1}\}])^{(\mathbf{1/128})}//\mathbf{N}$
Out[3]= {1.87939,1.53209,0.347296}

Знаки корней многочлена можно определить, применяя критерии п.7, или непосредственной подстановкой. Так, например, подставим w[[1]]=1.87939 в исходный многочлен $Q[x]$, получим

In[4]:= $\mathbf{Q}[\mathbf{w}[[\mathbf{1}]]]$
Out[4]= 2
a
In[5]:= $\mathbf{Q}[-\mathbf{w}[[\mathbf{1}]]]$
Out[5]= -4.9027×10^{-13}

Значит, -1.87939 есть корень.

Случаи комплексных корней рассматриваются аналогично, см. [5].

n.7 Матричный метод

Так как приведенный многочлен

$$x^n + a_1 x^{n-1} + ... + a_{n-1} x + a_n \tag{1}$$

является характеристическим для матрицы

$$m = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ . & . & . & . & . & . \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ -a_n & -a_{n-1} & -a_{n-2} & \cdots & -a_2 & -a_1 \end{pmatrix}, \tag{2}$$

то корни многочлена (1) являются собственными значения матрицы (2). Для определения собственных значений матрицы (2) можно применить, например, QR алгоритм (§12, n.5).

Рассмотрим пример. Требуется найти корни следующего приведенного многочлена:

In[1]:= $\mathbf{P[x\_] = x^8 - x^2 + 2x - 3}$

Введем степень многочлена.

In[3]:= $\mathbf{n = 8}$

Построим матрицу (2) по данному многочлену. Для этого определим нулевую матрицу a размерности $n = 8$, изменим на 1 ее наддиагональные элементы и заменим последнюю строку на список коэффициентов многочлена $P[x]$, без коэффициента при x^n и введем копию матрицы a, которая нам потребуется для проверки результата:

```
In[4]:= a = Table[0, {i, 1, n}, {i, 1, n}];
        Do[a[[i, i + 1]] = 1, {i, 1, n - 1}];
        a[[n]] = -Drop[CoefficientList[P[x], x], -1];
        a1 = a
```

Out[4]=

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 3 & -2 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Матрица a почти треугольная, так что к ней не надо применять метод Гивенса. Запускаем QR алгоритм для матрицы a:

```
In[5]:= t[a_, i_, j_] := Block[{c, s},
                  t1 = IdentityMatrix[n];
                  c = a[[i, i]]/Sqrt[a[[i, i]]^2 + a[[i, j]]^2]//N;
                  s = a[[i, j]]/Sqrt[a[[i, i]]^2 + a[[i, j]]^2]//N;
                  t1[[i, i]] = t1[[j, j]] = c;
                  t1[[i, j]] = -s;
                  t1[[j, i]] = s; t1]
        f = 0;
        While[
          Sum_{i=1}^{3} Abs[a[[i, i + 1]] a[[i + 1, i + 2]]] > 0.01,
          f = f + 1;
          q = IdentityMatrix[n];
          Do[
            tq = t[a, i, i + 1];
            q = q.tq;
            a = a.tq//Chop,
          {i, 1, n - 1}];
          a = Transpose[q].a;
        ];
        f
```

Цикл проработал

Out[5]= 240

раз. Матрица a примет следующий вид:

In[6]:= $\mathbf{a//Chop}$

Out[6]=

$$\begin{pmatrix}
-1.279815 & 0.000523 & 0 & 0 & 0 & 0 & 0 & 0 \\
-0.768299 & -1.081746 & 0.860601 & 0 & 0 & 0 & 0 & 0 \\
-0.687249 & -1.129552 & -0.517913 & 2.083014\times10^{-10} & 0 & 0 & 0 & 0 \\
-0.350381 & -0.368516 & 0.032982 & -0.042980 & 1.025115 & 0 & 0 & 0 \\
-1.305325 & -1.064950 & 0.368773 & -1.233141 & 0.296088 & 0.008683 & 0 & 0 \\
0.999300 & 0.816287 & -0.281511 & 0.326769 & -0.252041 & 1.090802 & 0.000345 & 0 \\
-0.432635 & -0.330532 & 0.140139 & -0.091859 & 0.122801 & -0.050060 & 0.741533 & 0.670214 \\
1.297703 & 1.052921 & -0.371260 & 0.401327 & -0.331881 & 0.220516 & -0.789619 & 0.794030
\end{pmatrix}$$

Вводим команды, определяющие собственные значения матрицы a (см. §12, п.5).:

In[7]:=
```
vsp[a_, i_] := Block[{q, d},
    q = Solve[(a[[i, i]] - x)(a[[i + 1, i + 1]] - x) -
        a[[i, i + 1]]a[[i + 1, i]] == 0, x];
    d = If[Abs[q[[2, 1, 2]]] <= Abs[q[[1, 1, 2]]], x /. q, Reverse[x /. q]];
    rt = Append[rt, d]//Flatten;]
```

In[8]:=
```
korni[a_] := Block[{rt = {}, i = 1, q, d},
    While[i < Length[a] - 1,
      If[Abs[a[[i, i + 1]]] <= Abs[a[[i + 1, i + 2]]],
        rt = Append[rt, a[[i, i]]]//Flatten; i = i + 1,
        vsp[a, i]; i = i + 2]
    ];
    If[i == Length[a] - 1, vsp[a, i]];
    If[i == Length[a], rt = Append[rt, a[[n, n]]]]; rt]
```

Находим собственные значения матрицы a и корни исходного многочлена:

In[9]:=
```
korni[a]//N
```

Out[9]=

$\{-1.27982, -0.79983 - 0.944784\,i, -0.79983 + 0.944784\,i,$
$0.126554 - 1.11147\,i, 0.126554 + 1.11147\,i, 1.0908,$
$0.767782 - 0.726998\,i, 0.767782 + 0.726998\,i\}$

Для сравнения воспользуемся встроенной в программу Mathematica функцией:

In[10]:=
```
Eigenvalues[a1]//N
```

Out[11]=

$\{-1.27982, -0.799828 - 0.944998\,i, -0.799828 + 0.944998\,i,$
$0.126555 - 1.11254\,i, 0.126555 + 1.11254\,i, 1.09107,$
$0.767746 - 0.727026\,i, 0.767746 + 0.727026\,i\}$

Замечание. Для многочлена

In[1]:=
```
P[x_] = x^8 + 1
```

данный способ не позволит найти его корни, так как модули всех корней равны 1. Сделаем сдвиг спектра матрицы a, добавим к 4-ой группе команд команду

```
a = a - λ IdentityMatrix[n]
```

где λ некоторое число, например, 1. Находим собственные значения измененной матрицы и совершаем обратный сдвиг спектра - вместо команды 9 выполним команду

In[9]:=
```
(korni[a]//N) + λ
```

Задание. Составить стандартную команду определяющую корни многочлена.

§15 Решение систем нелинейных уравнений

Рассмотрим итерационные методы решения систем нелинейных уравнений вида:

$$\begin{cases} f_1(x_1, x_2, ..., x_n) = 0, \\ f_2(x_1, x_2, ..., a_n) = 0, \\ \quad . \quad . \quad . \\ f_n(x_1, x_2, ..., a_n) = 0. \end{cases} \tag{1}$$

Запишем систему (1) в матричном виде.

Пусть $x = (x_1, x_2, ..., x_n)$. Функции в левой части (1) коротко можно записать так $f_1(x), ..., f_n(x)$. Пусть $F(x) = (f_1(x), f_2(x), ..., f_n(x))$, $0 = (0, 0, ..., 0)$. Тогда систему (1) можно переписать в виде:

$$F(x) = 0. \tag{1'}$$

Допустим, что систему (1) можно привести к виду:

$$\begin{cases} x_1 = \varphi_1(x_1, x_2, ..., x_n), \\ x_2 = \varphi_2(x_1, x_2, ..., a_n), \\ \quad . \quad . \quad . \\ x_n = \varphi_n(x_1, x_2, ..., a_n) \end{cases} \tag{2}$$

или в матричной форме:

$$x = \Phi(x), \tag{2'}$$

где $\Phi(x) = (\varphi_1(x), \varphi_2(x), ..., \varphi_n(x))$.

Уравнение (2′) позволяет построить последовательность векторов $\{x^{(n)}\}$, $n = 1, 2, ...$, по формуле:

$$x^{(n)} = \Phi(x^{(n-1)}), \tag{3}$$

начиная с некоторого начального приближения $x^{(0)}$. Если функция $\Phi(x)$ непрерывна, то предел построенной последовательности будет и решением системы уравнений (2), а, значит, и (1). При сходимости такой последовательности, всегда найдется член последовательности, который можно рассматривать в качестве решения системы с заданной точностью ε. Такой способ приближенного определения корня системы называется итерационным методом.

Следующая теорема показывает при каких условиях последовательность (3) сходится к решению.

Теорема. Пусть вектор-функция $\Phi(x)$ векторного аргумента дифференцируема в замкнутом шаре $S(y^{(0)}, r)$ с центром в точке $y^{(0)}$ и радиусом r. Если некоторая норма якобиана $\Phi'(x)$ функции $\Phi(x)$ удовлетворяет в шаре $S(y^{(0)}, r)$ условию:

$$||\Phi'(x)|| \leqslant q < 1 \tag{4}$$

при некотором q и

$$||y^{(0)} - \Phi(y^{(0)})|| \leqslant r(1 - q), \tag{5}$$

то последовательность, построенная по формуле (3) с начальным приближением $x^{(0)} \in S(y^{(0)}, r)$, сходится к единственному решению ξ уравнения (2) и выполняется неравенство:

$$||\xi - x^{(n)}|| \leqslant \frac{q}{1 - q}||x^{(n)} - x^{(n-1)}||. \tag{6}$$

Метод Ньютона. При решении систем вида (1) применяют так же итерационный метод Ньютона, который характеризуется способом приведения уравнения (1′) к виду (2′),

Пусть

$$\xi = x + \Delta x. \tag{7}$$

Применяя формулу Тейлора, запишем

$$0 = F(\xi) = F(x + \Delta x) = F(x) + F'(x)\Delta x + 0(||\Delta x||),$$

где $F'(x)$ - якобиан функции $F(x)$. Пренебрегая бесконечно малыми, получим уравнение

$$F(x) + F'(x)\Delta x = 0.$$

Отсюда, допуская, что якобиан $F'(x)$ обратим, найдем приращение

$$\Delta x = -[F'(x)]^{-1}.F(x).$$

Если поставим найденное приращение в (7), то получим второе приближение $x^{(1)}$ решения

$$x^{(1)} = x - [F'(x)]^{-1}.F(x).$$

Для вектора $x^{(1)}$ повторим рассуждение, получим вектор $x^{(2)}$ и так далее. В результате получим последовательность векторов $\{x^{(n)}\}$ такую, что

$$x^{(n)} = x^{(n-1)} - [F'(x^{(n-1)})]^{-1}.F(x^{(n-1)}), \tag{8}$$

$n = 1, 2, ...$, $x^{(0)} = x$.

Легко проследить аналогию многомерного случая (8) с методом Ньютона для уравнением с одной переменной.

Можно доказать, что сходимость полученной итерационной последовательности к решению ξ - квадратичная:

$$||\xi - x_n|| \leqslant ||x_n - x_{n-1}||^2.$$

При построении итерационной последовательности по методу Ньютона (формула (8)) необходимо вычислять обратную матрицу для якобиана на каждом шаге. Рассмотрим вариации метода Ньютона, позволяющие не вычислять обратную матрицу якобиана.

Преобразуем (8) следующим образом

$$F'(x^{(n-1)}).(x^{(n)} - x^{(n-1)}) = F(x^{(n-1)}).$$

Теперь для построения элемента итерационной последовательности $x^{(n)}$ требуется решить систему линейных алгебраических уравнений

$$F'(x^{(n-1)}).p^{(n)} = F(x^{(n-1)}).$$

относительно $p^{(n)}$. Тогда $x^{(n)} = x^{(n-1)} + p^{(n)}$.

Еще одна модификация метода Ньютона содержится в формуле

$$x^{(n)} = x^{(n-1)} - [F'(x^{(0)})]^{-1}.F(x^{(n-1)}).$$

Здесь обратная матрица $[F'(x^{(0)})]^{-1}$ вычисляется один раз в начальной точке.

Найдем решение следующей нелинейной системы уравнений методом итераций и методом Ньютона:

$$\begin{cases} Sin[\frac{x}{3} - 1] + y - 0.1 = 0, \\ x + Cos[\frac{y}{2}] - 1 = 0 \end{cases} \tag{9}$$

с точностью $\varepsilon = 0.000001$.

Проведем изоляцию корня графически, то есть, найдем прямоугольную область

$$D : \begin{cases} a_1 \leqslant x \leqslant a_2, \\ b_1 \leqslant y \leqslant b_2, \end{cases}$$

в которой расположено единственное решение системы (9).

Введем функции - левые части уравнений (9):

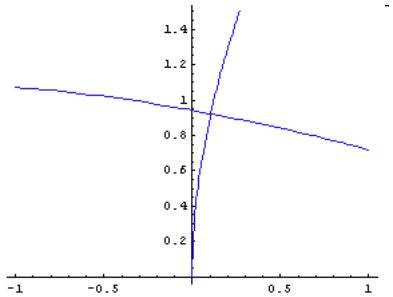
In[1]:= $\mathbf{f[x\_, y\_] := Sin[x/3 - 1] + y - 0.1}$

$\mathbf{g[x\_, y\_] := x + Cos[y/2] - 1}$

и построим множества, заданные уравнениями системы (9). Варьируя интервалы изменения переменных x и y в команде ContourPlot для получения наглядного пересечения графиков, можно прийти к следующей команде[1]:

In[2]:= $\mathbf{ContourPlot[\{f[x, y] == 0, g[x, y] == 0\}, \{x, -1, 1\}, \{y, 0, 1.4\}]}$

Out[2]=



Из графика видно, что область изоляции корня имеет вид:

$$D : \begin{cases} 0 \leqslant x \leqslant 0.3, \\ 0.8 \leqslant y \leqslant 1. \end{cases}$$

Начальное приближение нужно брать вблизи решения системы, то есть точки пересечения построенных графиков. Глядя на график, выберем в качестве начального приближения $\{x_0, y_0\} = \{0, 1\}$.

Далее, в качестве нормы вектора (матрицы) будем рассматривать норму $||t||_\infty$, ($||a||_1$). То есть, норму вектора t с координатами $\{x, y\}$ и норму матрицы $a = (a_{i,j})$ порядка n будем вычислять по формулам:

$$||t|| = max(|x|, |y|), \qquad ||a|| = max\Big(\sum_{j=1}^{n} |a_{1,j}|, ..., \sum_{j=1}^{n} |a_{n,j}|\Big)$$

[1] В программе Mathematica 5 вместо ContourPlot можно применить команду ImplicitPlot, предварительно загрузив пакет командой $<< Graphics`ImplicitPlot`$.

Эти нормы можно реализовать командми:

In[3]:= $\mathbf{nrv[t\_] := Max[Abs[t[[1]]], Abs[t[[2]]]]}$

$$\mathbf{nrm[t\_] := Max[Table[\sum_{j=1}^{Length[t]} Abs[t[[i,j]]], \{i,1,Length[t]\}]]}$$

Две последние команды можно объединить в одну:

In[4]:=
```
nr[t_] := Block[{w, n = Length[t]},
                If[VectorQ[t],
                w = Max[Abs[t[[1]]], Abs[t[[2]]]],
                w = Max[Table[Sum[Abs[t[[j, i]]], {i, 1, n}], {j, 1, n}]]]; w]
```

Здесь команда $VectorQ[t]$ возвращает True, если t - вектор и False в противном случае.

n.1 Метод итераций

Приведем уравнения системы (9) к виду (2), разрешая, например, первое уравнение системы (9) относительно y, второе - относительно x:

$$x = -Cos[\frac{y}{2}] + 1, \qquad y = -Sin[\frac{x}{3} - 1] + 0.1.$$

Если для полученной системы условия теоремы не будут выполняться, то уравнения (9) следует привести к виду (2) другим способом.

Проверка условия теоремы. Проверим выполнение условия (4) теоремы. Введем функцию:

In[5]:= $\mathbf{\Phi[x\_, y\_] := \{-Cos[y/2] + 1, -Sin[x/3 - 1] + 0.1\}}$

Очистим значения переменных перед дифференцированием по этим переменным:

In[6]:= $\mathbf{Clear[x, y]}$

и найдем якобиан функции $\mathbf{\Phi[x, y]}$:

In[7]:= $\mathbf{Jac[\{x\_, y\_\}] = Transpose[\{D[\Phi[x, y], x], D[\Phi[x, y], y]\}]}$

Out[7]= $\{\{0, \frac{1}{2}Sin[\frac{y}{2}]\}, \{-\frac{1}{2}Cos[1 - \frac{x}{2}], 0\}\}$

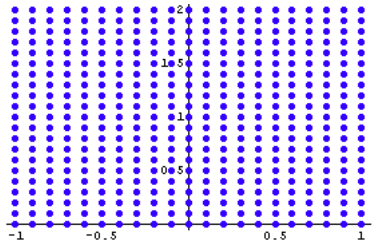
Вводим начальное приближение $\{x_0, y_0\}$:

In[8]:= $\mathbf{x_0 = 0; y_0 = 1;}$

Возьмем некоторое r>0, например, r=1 и рассмотрим квадрат $x_0 - r \leqslant x \leqslant x_0 + r$, $y_0 - r \leqslant y \leqslant y_0 + r$. Такой квадрат является кругом радиуса r в метрике определенной нормой nrv[t]. Для оценки нормы якобиана $nrm[Jac[x, y]]$ в таком круге покроем круг множеством точек с шагом 0.1. Нарисуем для наглядности такой круг с множеством точек:

In[9]:=
```
r = 1;
pnt = Partition[Table[{x, y}, {x, x0 - r, x0 + r, .1},
{y, y0 - r, y0 + r, .1}]//Flatten, 2];
Show[Graphics[{Hue[0.7], PointSize[0.02], Point/@pnt}],
Axes → True]
```

Out[9]=



Найдем значение нормы якобиана в каждой точке и выберем максимальное значение:

In[10]:= $\mathbf{r = 1;}$
$\mathbf{q = Max[Table[nrm[Jac[\{x, y\}]], \{x, x_0 - r, x_0 + r, .1\},}$
$\mathbf{\{y, y_0 - r, y_0 + r, 0.1\}]//Flatten]}$

Out[10]= 0.43879

Таким образом, условие (4) выполняется с $q \approx 0.45$. Проверим условие (5) теоремы. Для этого вычислим

In[11]:= $\mathbf{a = nr[\{x_0, y_0\} - F[x_0, y_0]]}$

Out[11]= 0.122417

Отсюда получаем:

In[12]:= $\mathbf{a < r(1 - q)}$

Out[12]= True

Все условия теоремы выполняются.

Решение системы. Теперь можно начинать строить итерационную последовательность. Первый элемент последовательности это начальное приближение.

In[13]:= $\mathbf{\{x, y\} = \{x_0, y_0\}}$

Найдем второй элемент итерационной последовательности. Запомним первый элемент последовательности (начало программы):

In[14]:= $\mathbf{z = \{x, y\}}$

Out[14]= {0,1}

Выполним команды:

In[15]:= $\mathbf{x = -Cos[\frac{y}{2}] + 1}$
$\mathbf{y = -Sin[\frac{x}{3} - 1] + 0.1}$

получим:

Out[15]= 0.122417

Out[16]= 0.918729

Следовательно, второй элемент последовательности
$\{x, y\} = \{0.122417, 0.918729\}$.

Проверяем условия (6) остановки процесса нахождения элементов итерационной последовательности:

In[17]:= $\mathbf{\frac{q}{(1 - q)} nr[z - \{x, y\}] < 0.000001}$

Out[17]= False

Условие (6) не выполняется. Переходим на начало программы, запоминаем второй элемент последовательности, находим третий элемент, проверяем условие остановки и так далее, до тех пор, пока условие остановки не вернет True.

Применим команду **Do** и объединим все команды в цикл. При этом считаем, что 100 шагов цикла нам хватит. Команда Break прервет цикл при достижении данной точности и вывода ответа на экран.

In[18]:=
```
{x, y} = {0, 1};
Do[z = {x, y};
 y = -Sin[x/3 - 1] + 0.1;
 x = -Cos[y/2] + 1;
 If[q/(1-q) nr[z - {x, y}] < 0.000001,
 Print["Решение системы"{x, y}, "получено на", i, "шаге"];
 Break[]],
{i, 1, 100}]
```
Out[18]= Решение системы {0.104427, 0.922158} получено на 9 шаге с погрешность 0.000001.

n.2 Метод Ньютона

Как уже отмечалось, метод Ньютона отличается от метода итераций способом приведения уравнения (1) к виду (2). Для метода Ньютона формула (2) имеет вид (8). Составим формулу (8) для поставленной задачи.

Очистим переменные перед дифференцированием. Считаем, что первые четыре команды выше - выполнены.

In[5]:= `Clear[x, y]`

и рассмотрим функцию:

In[6]:= `F[x_, y_] := {f[x, y], g[x, y]}`

Введем якобиан функции **F**[**x**, **y**]:

In[7]:= `FJac[x_, y_] = Transpose[{D[F[x, y], x], D[F[x, y], y]}]`

Out[7]= $\{\{\frac{1}{3}Cos[1-\frac{x}{3}], 1\}, \{1, -\frac{1}{2}Sin[\frac{y}{2}]\}\}$

Введем функцию:

In[8]:= `Λ[x_, y_] = -Inverse[FJac[x, y]]`

Следовательно формула (8) примет вид

$$\{x, y\} = \{x, y\} + \Lambda[x, y].F[x, y] \qquad (10).$$

Отметим, что это уравнение равносильно уравнению (1').

Проверка условий теоремы для правой части (10) аналогична предыдущему случаю.

Найдем якобиан функции $\{x, y\} + \Lambda[x, y].F[x, y]$.

In[9]:=
```
Jac[x_, y_] = Transpose[{D[{x, y} + Λ[x, y].F[x, y], x],
                         D[{x, y} + Λ[x, y].F[x, y], y]}]
```

Возьмем r=1 и оценим максимум нормы **Jac**[**x**, **y**] в квадрате $x_0 - r \leqslant x \leqslant x_0 + r$, $y_0 - r \leqslant y \leqslant y_0 + r$.

In[10]:=
```
r = 1;
q = Max[Table[nr[Jac[{x, y}]], {x, .x0 - r, x0 + r, 0.1},
              {y, y0 - r, y0 + r, 0.1}]//Flatten]
```
Out[10]= 0.248381

Таким образом $q = 0.248381$.

Вводим начальное приближение

In[11]:= `{x, y} = {0, 1};`

и проверяем условие (5):

In[12]:= `a = nr[{x, y} - {x, y} - Λ[x, y].F[x, y]]`

```
Out[12]= 0.103902
In[13]:= a < r(1 − q)
Out[13]= True
```

Условия теоремы выполняются.

Решение системы. Формируем цикл.

```
In[14]:= {x, y} = {0, 1};
          Do[z = {x, y};
             {x, y} = {x, y} + Λ[x, y].F[x, y]//N;
             If[q/(1 − q)nr[z − {x, y}] < 0.000001,
             Print["Решение системы"{x, y}, "получено на", i, "шаге."];
             Break[]],
           {i, 1, 100}]
Out[14]= Решение системы {0.104427, 0.922158} получено на 3 шаге.
```

Погрешность решения равна 0.000001!

n.3 Упрощенный метод Ньютона

В продолжении предыдущего пункта приведем упрощенный метод Ньютона. Обратная матрица $\Lambda[x, y]$ вычисляется только один раз, в начальном приближении.

```
In[16]:= {x, y} = {0, 1};
          Q = Λ[0, 1];
         Do[z = {x, y};
          {x, y} = {x, y} + Q.F[x, y]//N;
          If[q/(1 − q)nr[z − {x, y}] < 0.000001,
          Print["Решение системы"{x, y}, "получено на",
          i, "шаге."]; Break[]],
         {i, 1, 100}]
Out[16]= Решение системы {0.104427, 0.922158} получено на 3 шаге.
```

§16 Метод градиентного спуска

Метод градиентного спуска - это итерационный метод нахождения решения системы нелинейных уравнений. Этим же методом можно найти локальный минимум функции многих переменных. Рассмотрим применение метода к системе двух нелинейных уравнений с двумя неизвестными. Многомерный случай рассматривается аналогично.

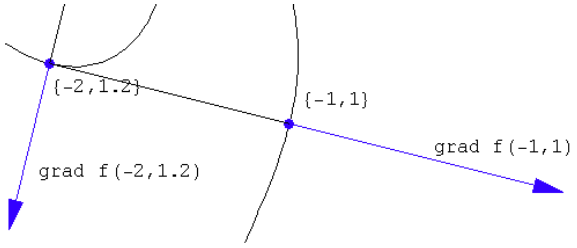
Для решения системы уравнений:

$$\begin{cases} f_1(x, y) = 0, \\ f_2(x, y) = 0 \end{cases} \tag{1}$$

методом градиентного спуска рассмотрим функцию:

$$f(x, y) = f_1(x, y)^2 + f_2(x, y)^2$$

Ноль функции $f(x, y)$ является решением системы (1). Найдем локальный минимум функции $f(x, y)$. То есть решим задачу нахождения локального минимума функции. Если найденный минимум будет и нулем функции $f(x, y)$, то мы получим и решение системы (1).



К минимальной точке функции двигаемся по ломаной начиная с произвольной точки, например, $(-1, 1)$. Построим первое звено ломаной. Проведем через точку $(-1, 1)$ прямую l, параллельную градиенту

$$grad\ f(-1,1) = \{\frac{\partial f}{\partial x}(-1,1), \frac{\partial f}{\partial y}(-1,1)\}$$

функции $f(x, y)$ в точке $(-1, 1)$ и найдем минимум $g(t) = f(x, y)|_l$ сужения функции $f(x, y)$ на прямую l. При параметрическом задании прямой l:

$$x = -1 - t\,\frac{\partial f}{\partial x}(-1,1), \quad y = 1 + t\,\frac{\partial f}{\partial y}(-1,1)$$

функция $g(t)$ примет вид

$$g(t) = f(-1 - t\frac{\partial f}{\partial x}(-1,1), 1 - t\frac{\partial f}{\partial y}(-1,1)).$$

Для определения точек локального минимума функции $g(t)$ надо решить уравнение $g'(t) = 0$. Минимальное и положительное значение t решения уравнения $g'(t) = 0$ будет соответствовать точке касания прямой l с нижележащим множеством уровня функции $f(x, y)$. Пусть эта точка касания имеет координаты $(-2, 1.2)$. Отрезок с концами в точках $(-1, 1)$ и $(-2, 1.2)$ и есть первое звено ломаной. Отметим, что значения функции $f(x, y)$ уменьшаются при движении по прямой l от точки $(-1, 1)$ к точке $(-2, 1.2)$. Аналогично строим второе звено ломаной, начиная с точки $(-2, 1.2)$. Вершина ломанной (x_n, y_n), удовлетворяющая условию

$$max(|x_n - x_{n-1}|, |y_n - y_{n-1}|) < \varepsilon,$$

будет являться минимальным значением функции $f(x, y)$ с данной погрешностью ε.

Решаем конкретную задачу. Найдем решение системы уравнений

$$\begin{cases} x + y^2 - 2 = 0, \\ y - x^2 + 3 = 0 \end{cases} \tag{2}$$

методом градиентного спуска.

Очистим переменные перед определением функций и дифференцированием по этим переменным:

In[1]:= $\mathbf{Clear[x, y, t]}$

Определим функцию для минимизации:

In[2]:= $\mathbf{f[\{x\_, y\_\}] = (x + y^2 - 2)^2 + (y - x^2 + 3)^2}$

и найдем ее градиент:

In[3]:= $\mathbf{gradf[x\_, y\_] = \{D[f[\{x, y\}], x], D[f[\{x, y\}], y]\}}$

Out[3]= $\{-4x(3 - x^2 + y) + 2(-2 + x + y^2), 2(3 - x^2 + y) + 4y(-2 + x + y^2)\}$

Рассмотрим сужение $g[t]$ функции $f[\{x, y\}]$ на прямую, проходящую через точку с координатами (x, y) параллельно градиенту функции $f[\{x, y\}]$:

In[4]:= $\mathbf{g[t\_] = f[\{x, y\} - t\ gradf[x, y]]//N//Chop}$

Out[4]= $((3.) + y - 1.t(2.(3.) - 1.x^2 + y) + 4.y(-2. + x + y^2)) - 1.(x - 1.t(-4.x(3.) - 1.x^2 + y) + 2.(-2. + x + y^2)))^2)^2 + (-2. + x - 1.t(-4.x(3.) - 1.x^2 + y) + 2.(-2. + x + y^2)) + (y - 1.t\ (2.(3.) - 1.x^2 + y) + 4.y\ (-2. + x + y^2)))^2)^2$

Возьмем начальную точку ломаной. Такая точка берется произвольно, но от ее выбора зависит к какой точке минимума мы подойдем по построенной ломаной.

In[5]:= $\mathbf{\{x, y\} = \{-1, 1\}}$;

Следующая встроенная команда находит стационарные точки функции $g(t)$, решая уравнение $g'(t) = 0$.

In[6]:= $\mathbf{s = Solve[g'[t] == 0, t]}$

Out[6]= $\{\{t- > -0.352591\}, \{t- > -0.093639\}, \{t- > 0.125218\}\}$

Решение s может иметь несколько стационарных точек, при этом, некоторые из них могут быть комплексными. Число действительных решений равно числу точек касания прямой, проходящей параллельно градиенту, с множествами уровня. Точке минимума будут соответствовать положительные решения.

Для выбора действительного и положительного t из списка s применим функцию $Select$ с проверкой условия $((Im[\#[[1, 2]]] == 0)\&\&Re[\#[[1, 2]]] > 0)\&$. В это условие вместо решетки # подставляются по очереди элементы списка s. Тогда $\#[[1, 2]]$ - числовые элементы списка s. Если мнимая часть числа $\#[[1, 2]]$ равна 0 и число является положительным, то такой элемент списка выбирается. Из выбранного списка берется первый элемент командой $First$ и присваивается снова переменной s. В нашем случае, переменная s примет вид $s = \{t \to 0.125218\}$.

In[7]:= $\mathbf{s = Select[s, ((Im[\#[[1, 2]]] == 0)\&\&Re[\#[[1, 2]]] > 0)\&][[1]]}$

Out[7]= $\{t \to 0.125218\}$

Запоминаем текущую вершину ломаной

In[8]:= $\mathbf{z = \{x, y\}}$

и находим новую. Здесь, вместо t будет подставлено значение из списка s.

In[9]:= $\mathbf{\{x, y\} = (\{x, y\} - t\ gradf[x, y]/.s//N)}$

Out[9]= {-2.00175,1.25044}

Проверяем условие остановки.

In[10]:= $\mathbf{Sqrt[(z - \{x, y\}).(z - \{x, y\})] < 0.0001}$

Out[10]= False

Так как условие остановки построения ломаной не выполняется, то повторим все команды, начиная с команды 6. Повторяем до тех пор, пока вывод 10 не будет True. Здесь уместно подключить команду цикла.

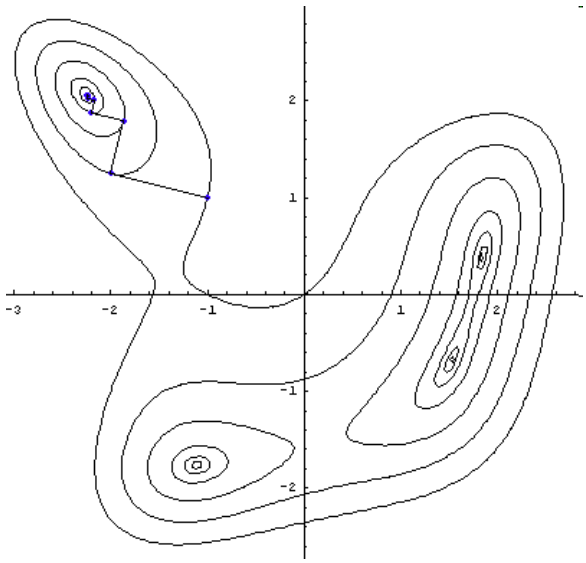
Кроме приведенных команд, в цикл добавим команды вычисление трех переменных, которые понадобятся для графической иллюстрации метода. Переменная *poi* будет содержать список координат вершин ломаной, список *ur* будет содержать значения функции $f[\{x,y\}]$ в вершинах ломаной. Эти значения нужны для построения множеств уровней функции, на которых лежат вершины ломаной. Переменная *zi* будет равна числу звеньев ломаной.

```
In[11]:= ur = {};
In[12]:= poi = {};
In[13]:= zi = 0;
```

Команды *Append* добавляют к спискам *poi* и *ur* координаты очередной вершины ломаной и значение функции в этой вершине. При первом проходе цикла $poi = \{-1,1\}$, а $ur = \{f[\{-1,1\}]\}$.

```
In[14]:= Do[poi = Append[poi, {x, y}];
            ur = Append[ur, f[{x, y}]];
            s = Solve[g'[t] == 0, t];
            s = First[Select[s, ((Im[#[[1, 2]]] == 0)&&Re[#[[1, 2]]] > 0)&]];
            z = {x, y};
            {x, y} = ({x, y} - t gradf[x, y]/.s//N);
            If[Sqrt[(z - {x, y}).(z - {x, y})] < 0.001,
              Print["Точка минимума", {x, y}, "получено на",
              zi = i, "шаге"]; Abort[]],
          {i, 1, 100}]
Out[14]= Точка минимума {-2.24977, 2.06146} получена на 13 шаге
```

Найденное минимальное значение можно считать решением системы уравнений так как $f[\{-2.24977, 2.06146\}] = 2.85614 \times 10^{-8}$.



Построим множества уровня функции $f[\{x,y\}]$, содержащие вершины ломаной и саму ломаную.

Очистим переменные перед построением графика функции:

```
In[15]:= Clear[x, y]
```

и составим список множеств уровня функции $\{\{x,y\}|f[\{x,y\}] = ur[[i]]\}$, где i меняется от 1 до zi=13 - числа звеньев ломаной.

```
In[16]:= gr = Table[
            ContourPlot[(x + y^2 - 2)^2 + (y - x^2 + 3)^2 == ur[[i]],
              {x, -5, 5}, {y, -5, 5},
              PlotPoints → {200, 200}],
            {i, 1, zi}];
```

Сформируем список вершин ломаной:

```
In[17]:= gr1 = Graphics[{Hue[0.7], PointSize[0.01],
                 Point[#]}]&/@poi;
```

список звеньев ломаной:

```
In[18]:= tb = Table[{poi[[i]], poi[[i + 1]]}, {i, 1, zi - 1}];
```

построим ломаную:

```
In[19]:= gr2 = Graphics[Line/@tb];
```

и выведем все графики на экран:

```
In[20]:= Show[gr, gr1, gr2]
```

n.1 Разновидность метода градиентного спуска

Решение уравнения с одной переменной t, вообще говоря, не простая задача. Надо провести программно изоляцию корней и найти их, например, методом итераций или Ньютона, методом дихотомии или золотого сечения (см. Численные методы оптимизации). В предыдущей задаче для нахождения корней уравнения мы использовали встроенную удобную функцию Solve. Упрощение метода градиентного спуска заключается в том, что параметр t не вычисляется в каждой вершине ломаной, а выбирается постоянным. При этом на каждом шаге проверяется условие $f(x_n, y_n) < f(x_{n-1}, y_{n-1})$. Если это условие монотонности нарушается, то первоначальное значение t уменьшается до тех пор, пока условие монотонности не восстановится.

Более подробно, выбор t и возможные случаи можно посмотреть, например, в [2].

Например, в предыдущем способе минимизации функции положим $t = 0.01$ для всех вершин ломаной (дробления t не производится). Число звеньев ломаной при этом увеличится до 45. Ниже приведена программа с соответствующими изменениями и приведен график процесса минимизации.

```
In[1]:= Clear[x, y, t]
In[2]:= f[{x_, y_}] = (x + y^2 - 2)^2 + (y - x^2 + 3)^2
In[3]:= gradf[x_, y_] = {D[f[{x, y}], x], D[f[{x, y}], y]; }
In[4]:= g[t_] = f[{x, y} - t gradf[x, y]]//N//Chop;
In[5]:= ur = {};
In[6]:= poi = {};
In[7]:= zi = 0;
In[8]:= {x, y} = {-1, 1};
In[9]:= Do[poi = Append[poi, {x, y}];
        ur = Append[ur, f[{x, y}]];
        z = {x, y};
(* считаем, что параметр t=0.01 для всех звеньев ломаной *)
```

```
{x, y} = ({x, y} − 0.01gradf[x, y]//N);
If[Sqrt[(z − {x, y}).(z − {x, y})] < 0.001,
Print["Точка минимума", {x, y}, "получена на", zi = i,
"шаге"]; Abort[]],
{i, 1, 100}]
```

Out[9]= Точка минимума {-2.24958, 2.06126} получена на 45 шаге

Построим ломаную и множества уровня для этого случая. Загрузим пакет построения множеств, заданных неявно, и очистим переменные перед дифференцированием:

In[10]:=
```
Clear[x, y]
```

Строим множества уровня:

In[11]:=
```
gr = Table[ContourPlot[f[{x, y}] = ur[[i]],
{x, −5, 5}, {y, −5, 5}, PlotPoints → {200, 200}],
{i, 1, zi}];
```

Строим последовательность точек:

In[12]:=
```
gr1 = Graphics[{Hue[0.7], PointSize[0.01],
Point[#]}]&/@poi;
```

Формируем звенья ломаной:

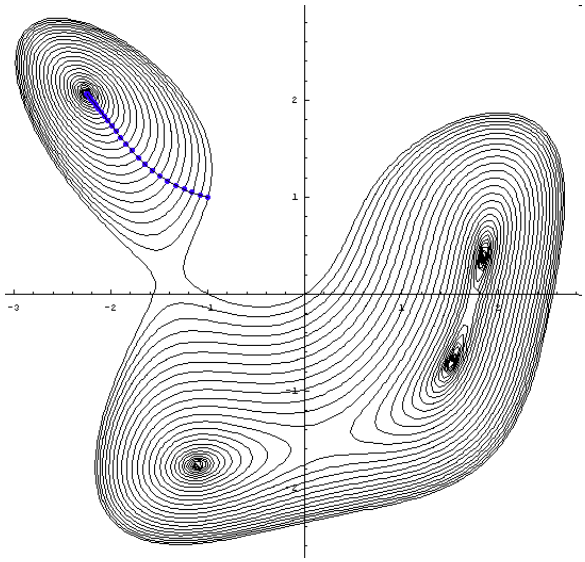
```
tb = Table[{poi[[i]], poi[[i + 1]]}, {i, 1, zi − 1}];
```

строим ломаную:

In[13]:=
```
gr2 = Graphics[Line/@tb];
```

и выводим все графики вместе:

In[14]:=
```
Show[gr, gr1, gr2]
```



§17 Аппроксимация и интерполяция функций

В этом параграфе рассматриваются интерполяционные многочлены Лагранжа, Чебышева, Ньютона для таблично заданных функций, аппроксимация функций методом наименьших квадратов и с помощью кубических сплайнов.

Под интерполяцией функции $f(x)$, определенной на множестве D, понимается доопределение функции $f(x)$ в точке $x \notin D$. Аппроксимация функции $f(x)$, заданной на множестве D, заключается в построении, как правило, более простой функции $g(x)$, заданной на множестве D и близкой к функции $f(x)$ в определенном смысле.

n.1 Многочлен Лагранжа

Пусть задана табличная функция $f = \{(x_i, y_i)\}$, $i = 0, 1, ..., n$. Интерполируем эту функцию на все точки числовой оси с помощью многочлена Лагранжа степени $k \leqslant n$.

Дадим определение. Для таблично заданной функции $f = \{(x_i, y_i)\}$, $i = 0, 1, ..., n$, интерполяционным многочленом Лагранжа называется многочлен $L(x)$ степени $k \leqslant n$ такой, что

$$L(x_i) = y_i \tag{1}$$

для всех $i = 0, 1, ..., n$.

Покажем, что многочлен Лагранжа для данной табличной функции f существует и единствен. Запишем многочлен степени $k \leqslant n$ в виде

$$L(x) = a_0 + a_1 x + ... + a_n x^n$$

и потребуем выполнение равенств (1):

$$\begin{array}{c} a_0 + a_1 x_0 + a_2 x_0^2 + ... + a_n x_0^n = y_0, \\ a_0 + a_1 x_1 + a_2 x_1^2 + ... + a_n x_1^n = y_1, \\ \cdot \qquad \cdot \qquad \cdot \\ a_0 + a_1 x_n + a_2 x_n^2 + ... + a_n x_n^n = y_n. \end{array} \tag{2}$$

Получилась система уравнений относительно коэффициентов a_0, $a_1, ...,$ a_n. Определитель этой системы есть определитель Вандермонда

$$\begin{vmatrix} 1 & x_0 & x_0^2 & ... & x_0^n \\ 1 & x_1 & x_1^2 & ... & x_1^n \\ . & . & . & ... & . \\ 1 & x_n & x_n^2 & ... & x_n^n \end{vmatrix}.$$

Как известно, определитель Вандермонда для попарно различных узлов x_0, $x_1, ...,$ x_n не равен нулю. Следовательно система (2) имеет единственное решение. Поэтому многочлен Лагранжа для данной табличной функции f существует и единствен. При этом, возможно, что некоторые коэффициенты будут равны нулю, что и может понизить степень многочлена Лагранжа до $k < n$.

Единственность многочлена Лагранжа позволяет утверждать, что многочлен степени $k \leqslant n$ вида

$$\begin{array}{r} L(x) = y_0 \dfrac{(x - x_1)(x - x_2)...(x - x_n)}{(x_0 - x_1)(x_0 - x_2)...(x_0 - x_n)} + \\ y_1 \dfrac{(x - x_0)(x - x_2)...(x - x_n)}{(x_1 - x_0)(x_1 - x_2)...(x_1 - x_n)} + ... \\ ... + y_n \dfrac{(x - x_0)(x - x_1)...(x - x_{n-1})}{(x_n - x_1)(x_n - x_2)...(x_n - x_{n-1})} \end{array} \tag{3}$$

или

$$L(x) = \sum_{j=0}^{n} y_j \prod_{\substack{i=0 \\ i \neq j}}^{n} \frac{x - x_i}{x_j - x_i}$$

и есть многочлен Лагранжа. Действительно, для многочлена (3) условия (1) выполняются и многочлен, стоящий справа в (3) имеет степень $k \leqslant n$.

Теперь можно интерполировать табличную функцию f в точку $\widetilde{x}$, доопределяя её в этой точке значением $L(\widetilde{x})$.

С помощью многочлена Лагранжа можно решать задачу аппроксимации функции $y = f(x)$, определенной на отрезке $[a, b]$. Для этого надо рассмотреть табличную функцию $g = \{(x_i, f(x_i))\}$, $i = 0, 1, ..., n$, с узлами $x_0 = a < x_1 < ... < x_n = b$ и для табличной функции построим многочлен Лагранжа $L(x)$. В некоторых случаев, замена более сложной функции $y = f(x)$ таким многочленом Лагранжа является достаточной.

Оценить погрешность многочлена Лагранжа можно только для достаточно хорошей функции и в следующем смысле.

Теорема. Пусть функция $f(x)$, определенная на отрезке $[a, b]$, $n+1$ раз дифференцируема, точки x_i, $i = 0, 1, ..., n$, такие, что $x_0 = a < x_1 < ... < x_n = b$. По табличной функции $g = \{(x_i, f(x_i))\}$, $i = 0, 1, ..., n$, построим многочлен Лагранжа $L(x)$. Тогда, для любой точки $\overline{x} \in [a, b]$,

$$f(\overline{x}) = L(\overline{x}) + \frac{f^{n+1}(\xi)}{(n+1)!}\omega(\overline{x}), \tag{4}$$

где $\xi = \xi(\overline{x}) \in [a, b]$, а $\omega(x) = (x - x_0)(x - x_1)...(x - x_n)$.

Доказательство. Для узлов x_i равенство (4) верно. Считаем, что точка $\overline{x}$ не является узлом. Рассмотрим функцию

$$u(x) = f(x) - L(x) - k\omega(x), \tag{5}$$

где k - постоянная. Пусть k такое, что $f(\overline{x}) - L(\overline{x}) - k\omega(\overline{x}) = 0$, то есть

$$k = \frac{f(\overline{x}) - L(\overline{x})}{\omega(\overline{x})}.$$

Найденное k подставим в (5) и получим, что функция $u(x)$ имеет $n + 2$ различных корня на отрезке $[a, b]$: $\overline{x}$ является корнем по выбору k, а $x_0, x_1, ..., x_n$ в силу определения многочлена Лагранжа. По теореме Ройля, производная $u'(x)$ будет иметь $n+1$ корень на отрезке $[a, b]$, вторая производная $u''(x)$ будет иметь n корней на отрезке $[a, b]$ и так далее, $n + 1$ производная $u^{(n+1)}(x)$ будет иметь 1 корень на отрезке $[a, b]$. Пусть этим корнем будет ξ: $u^{(n+1)}(\xi) = 0$. Отсюда и из (5) получим:

$$0 = u^{(n+1)}(\xi) = f^{(n+1)}(\xi) - L^{(n+1)}(\xi) - k\omega^{(n+1)}(\xi). \tag{6}$$

Так как многочлен $L(x)$ имеет степень $k \leqslant n$, то $L^{(n+1)}(x) = 0$. Для многочлена $\omega(x)$ степени $n + 1$ справедливо равенство $\omega^{(n+1)}(x) = (n + 1)!$. Поэтому (6) примет вид

$$f^{(n+1)}(\xi) - k(n + 1)! = 0.$$

Отсюда найдем k, подставим его (5), и найдем (5) при $x = \overline{x}$, в результате получим

$$0 = u(\overline{x}) = f(\overline{x}) - L(\overline{x}) - \frac{f^{n+1}(\xi)}{(n+1)!}\omega(\overline{x}),$$

что совпадает с утверждением теоремы. Теорема доказана.

Следствие. Из равенства (4) можно получить оценку аппроксимации функции $f(x)$ таким многочленом Лагранжа:

$$|f(x) - L(x)| \leqslant \frac{M_n}{(n+1)!} \max_{x\in[a,b]} |\omega(x)|, \tag{7}$$

где $M_n = \max_{x\in[a,b]} |f^{(n+1)}(x)|$.

Отметим, что в программе Математика построить произведение всех элементов $a_0, a_1, ..., a_{10}$ за исключением, скажем 2-го и 5-го, можно следующим образом

In[1]:= $\prod_{\mathbf{i=0}}^{\mathbf{10}} \mathbf{If}[\mathbf{i} == \mathbf{2} || \mathbf{i} == \mathbf{5}, \mathbf{1}, \mathbf{a_i}]$

В результате получится

Out[1]= $a_0a_1a_3a_4a_6a_7a_8a_9a_{10}$

Поэтому многочлен Лагранжа в программе Mathematica можно записать компактно так

$$L_n[x\_] = \sum_{j=0}^{n} f[x_j] \prod_{i=0}^{n} If[i == j, 1, \frac{x - x_i}{x_j - x_i}] \tag{8}$$

Подготовка задания. Возьмем функцию

In[2]:= $\mathbf{f}[\mathbf{x\_}] = \mathbf{Sin}[\mathbf{x}];$

и найдем ее значения в равноотстоящих точках (узлах) на отрезке $[-6, 6]$, шаг следования узлов $h = 2.4$. Введем узды.

In[3]:= $\mathbf{Do}[\mathbf{x_i} = -\mathbf{6} + \mathbf{2.4\ i}, \{\mathbf{i}, \mathbf{0}, \mathbf{5}\}]$

В результате получим таблично заданную функцию

In[4]:= $\mathbf{fs} = \mathbf{Table}[\{\mathbf{x_i}, \mathbf{f}[\mathbf{x_i}]\}, \{\mathbf{i}, \mathbf{0}, \mathbf{5}\}]//\mathbf{N}$

Out[4]=

{{-6, 0.2794},{-3.6, 0.4425}, {-1.2, -0.9321}, {1.2, 0.932}, {3.6, -0.4425},
{6, -0.2794}}

Теперь можно сформулировать задание: для таблично заданной функции **fs** построить многочлен Лагранжа, построить графики таблично заданной функции **fs**, многочлена Лагранжа и функции $f(x)$.

Построим график функции $f(x)$:

In[5]:= $\mathbf{gr1} = \mathbf{Plot}[\mathbf{f}[\mathbf{x}], \{\mathbf{x}, -\mathbf{6}, \mathbf{6}\}, \mathbf{PlotStyle} \to \{\mathbf{Hue}[\mathbf{0.3}]\}]$

Out[5]=

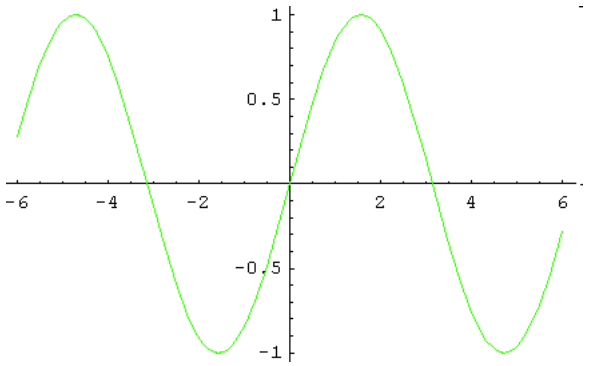
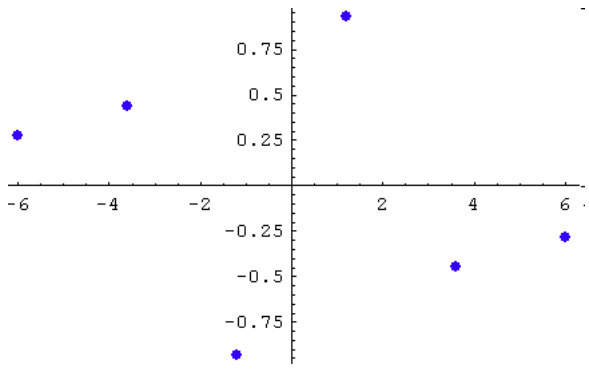


график таблично заданной функции

In[6]:= **gr2 = ListPlot[fs, PlotStyle → {Hue[0.7], PointSize[0.02]}]**

Out[6]=



Построим график многочлен Лагранжа. Сначала составим многочлен Лагранжа, воспользуемся формулой (8) при n=5:

In[7]:=

$$\mathbf{L_5[x\_] = \sum_{j=0}^{5} f[x_j] \prod_{i=0}^{5} If[i == j, 1, \frac{x - x_i}{x_j - x_i}]}$$

Out[7]=

-0.0000292424 (x - 6.) (x - 3.6) (x - 1.2) (x + 1.2) (x + 3.6) +
0.000975431 (x - 6.) (x - 3.6) (x - 1.2) (x + 6) (x + 3.6) +
0.000975431 (x - 6.) (x - 3.6) (x + 1.2) (x + 6) (x + 3.6) +
0.000231561 (x - 6.) (x - 1.2) (x + 1.2) (x + 6) (x + 3.6) -
0.0000292424 (x - 3.6) (x - 1.2) (x + 1.2) (x + 6) (x + 3.6) +
0.000231561 (x - 6.) (x - 3.6) (x - 1.2) (x + 1.2) (x + 6)

Для упрощения, выполним команду

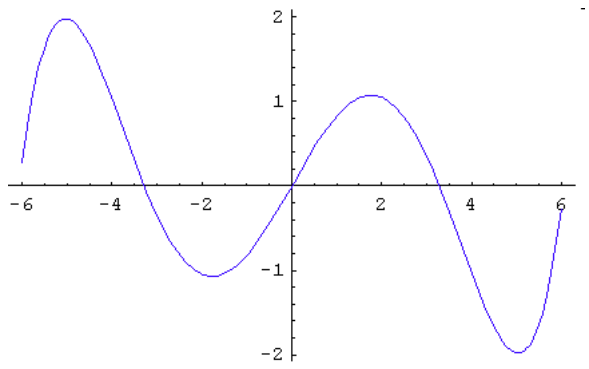
In[8]:= **%//N//Expand//Chop**

Out[8]= $0.00236\ x^5 - 0.112\ x^3 + 0.9331\ x$

Строим многочлен Лагранжа

In[9]:= **gr3 = Plot[L₅[x], {x, −6, 6}, PlotStyle → {Hue[0.7]}]**

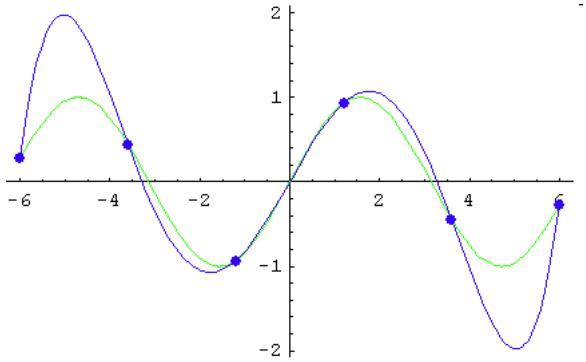
Out[9]=



Теперь совместим три графика для сравнения, получим

In[10]:= **Show[gr1, gr2, gr3]**

Out[10]=



Так построенный многочлен Лагранжа решает задачу аппроксимации для непрерывной функции $f(x)$. Найдем оценку такой аппроксимации, то есть оценим разность $|f(x)-L_5(x)|$ на отрезке $[-6,6]$. Для этого подсчитаем правую часть неравенства (7).

Пусть

In[11]:= $\omega[\mathbf{x\_}] := \prod_{\mathbf{i=0}}^{\mathbf{5}}(\mathbf{x}-\mathbf{x_i})$

Так как $M_5 = \max_{x\in[-6,6]}|Sin^{(6)}(x)| = 1$, то правая часть неравенства (2) равна $\max_{x\in[-6,6]}|\omega[x]/6!|$. Найдем этот максимум:

In[12]:= $\mathbf{Maximize}[\mathbf{Abs}[\omega[\mathbf{x}]]/\mathbf{6!}, \mathbf{x} \geqslant -\mathbf{6}\&\&\mathbf{x} \leqslant \mathbf{6}, \{\mathbf{x}\}]$

Out[12]= $\{1.34012, \{x \to 2.57688\}\}$

Таким образом, максимальное отклонение $|sin(x) - L_5(x)|$ равно 2.57688, с чем согласуются построенные графики.

n.2 Многочлены Чебышева и многочлен Лагранжа наилучшего приближения

Многочленом Чебышева называется функция

$$T(n,x) = cos(n\,arccos\ x), \tag{1}$$

где n-натуральное число или ноль, а $x \in [-1,1]$. При $n = 0,1$ получим $T(0,x) = 1$, $T(1,x) = x$.

Пусть $\alpha = arccos\ x$. Из тождества

$$\cos n\alpha = 2\cos\alpha\cos(n-1)\alpha - \cos(n-2)\alpha$$

получаем рекуррентную формулу

$$T(n,x) = 2xT(n-1,x) - T(n-2,x), \tag{2}$$

$n = 2,3,....$ Применяя (2), можно записать

$$\begin{aligned} T(2,x) &= 2xT(1,x) - T(0,x) = 2x^2 - 1,\\ T(3,x) &= 2xT(2,x) - T(1,x) = 2(2x^2-1) - x = 4x^3 - 3x,\\ T(4,x) &= 2xT(3,x) - T(2,x) = 8x^4 - 8x^2 + 1, ... \end{aligned} \tag{3}$$

Свойства многочленов Чебышева.

1. Из (3) видно, что функция (1) действительно является многочленом. При этом, при четном (нечетном) n многочлен Чебышева $T(n,x)$ содержит только четные (нечетные) степени x. Значит многочлен Чебышева есть четная или нечетная функция.

2. Старший коэффициент многочлена Чебышева $T(n,x)$ равен 2^{n-1}. Приведенный многочлен

$$\overline{T}(n,x) = \frac{1}{2^{n-1}} T(n,x).$$

3. Многочлен Чебышева $T(n,x)$ имеет n корней на отрезке $[-1,1]$:

$$x_k = \cos\frac{(2k+1)\pi}{2n}, \quad k = 0,1,...,n-1. \tag{4}$$

Доказательство. Если $\cos(n\arccos x) = 0$, то $n\arccos x = \frac{\pi}{2} + \pi k = \pi\frac{1+2k}{2}$, а $\arccos x = \pi\frac{1+2k}{2n} \in [0,\pi]$. Отсюда находим корни $x_k = \cos\frac{(2k+1)\pi}{2n}$. Найдем число корней. Решая неравенство $0 \leqslant \pi\frac{1+2k}{2n} \leqslant \pi$ получим, что $k = 0,1,...,n-1$.

4. Точки экстремума многочлена Чебышева

$$x_k = \cos\frac{\pi k}{n}, \quad k = 0,1,...,n.$$

чередуются с его корнями. При этом $T(n,x_k) = (-1)^k$ для всех k. Для приведенного многочлена Чебышева $\overline{T}(n,x_k) = \frac{(-1)^k}{2^{n-1}}$.

Доказательство. Точки экстремума определяются из уравнения $T'(n,x) = 0$ или $\sin(n\arccos x) = 0$. Отсюда $n\arccos x = \pi k$, где k - целое число. Но, требование $0 \leqslant \arccos x = \frac{\pi k}{n} \leqslant \pi$ приводит к значениям $k = 0,1,...,n$, а $x_k = \cos\frac{\pi k}{n}$. Отсюда $T(n,x_k) = \cos(n\arccos(\cos\frac{\pi k}{n})) = \cos(\pi k) = (-1)^k$. Из вида точек нулей и экстремумов следует их чередование.

5. Модуль максимального значения любого приведенного многочлена степени n на отрезке $[-1,1]$ не меньше модуля максимального значения приведенного многочлена Чебышева $T(n,x)$ на отрезке $[-1,1]$. Другими словами, приведенный многочлен Чебышева n-го порядка наименее уклоняется от нуля среди всех приведенных многочленов n-го порядка на отрезке $[-1,1]$.

Доказательство. Пусть $\overline{T}(n,x)$ приведенный многочлен Чебышева. Допустим противное, то есть существует приведенный многочлен $P(x)$ порядка n такой, что

$$\max_{x\in[-1,1]} |\overline{T}(n,x)| > \max_{x\in[-1,1]} |P(x)|. \tag{5}$$

Тогда многочлен $Q(x) = \overline{T}(n,x) - P(x)$ имеет порядок меньше n и в $n+1$ точке $x_k = \cos\frac{\pi k}{n}$, $k = 0,1,...,n$ в силу (5) принимает значения чередующихся знаков. Такой многочлен должен иметь n корней. Это противоречит тому, что многочлен $Q(x)$ порядка меньше n.

Построим теперь многочлен Лагранжа наилучшего приближения (аппроксимации) для функции $y = f(x)$ на отрезке $[a,b]$, применяя многочлены Чебышева. Узлы табличной функции, по которой строится многочлен Лагранжа, могут быть не равноотстоящими. Оказывается, с помощью многочленов Чебышева, можно так подобрать узлы, что многочлен Лагранжа, построенный по таким узлам, будет наилучшего приближения.

Пусть $x_0 = a < x_1 < ... < x_n = b$ точки отрезка $[a,b]$, а $\omega(x) = (x-x_0)(x-x_1)...(x-x_n)$. Зададим функцию $\omega(x)$ на отрезке $[-1,1]$, сделав замену переменной по формул:

$$x = \frac{b+a}{2} + \frac{b-a}{2}t,$$

где $t \in [-1,1]$. Если

$$x_i = \frac{b+a}{2} + \frac{b-a}{2} t_i, \tag{6}$$

$i = 0, 1, ..., n$, то

$$x - x_i = \frac{b-a}{2}(t - t_i),$$

следовательно, $\omega(x) = (x-x_0)(x-x_1)...(x-x_n) = (\frac{b-a}{2})^{n+1}(t-t_0)(t-t_1)...(t-t_n)$.

Если в качестве корней приведенного многочлена $q(t) = (t-t_0)(t-t_1)...(t-t_n)$ выберем корни многочлена Чебышева, то $q(t)$ - станет приведенным многочленом Чебышева степени $n+1$. По свойству (4)

$$\max_{t\in[-1,1]} |q(t)| = \frac{1}{2^n}.$$

Отсюда

$$\max_{x\in[a,b]} |\omega(x)| = \max_{t\in[-1,1]} |(\frac{b-a}{2})^{n+1}(t-t_0)(t-t_1)...(t-t_n)| = \frac{(b-a)^{n+1}}{2^{2n+1}}.$$

По табличной функции $(x_i, f(x_i))$, $i = 0, 1, ..., n$, где x_i найдены по формуле (6) через корни t_i многочлена Чебышева $(n+1)$ порядка (формула (4)), построим многочлен Лагранжа $L_n(x)$. Тогда погрешность аппроксимации функции $f(x)$ многочленом Лагранжа (см. следствие из теоремы п.1)

$$|f(x) - L_n(x)| \leqslant \frac{M_n}{(n+1)!} \max_{x\in[a,b]} |\omega(x)| = \frac{M_n}{(n+1)!} \frac{(b-a)^{n+1}}{2^{2n+1}}, \tag{7}$$

где $M_n = \max_{x\in[a,b]} |f^{(n+1)}(x)|$.

Если $a = -6$, $b = 6$, $f(x) = sin(x)$, $n = 5$, то оценка справа в (7) равна 2.025. Эта оценка не улучшаема (см., например, [2]).

В системе Mathematica рекуррентная формула (2) реализуется следующими командами:

In[1]:= $\mathbf{T[0, x] = 1}$

In[2]:= $\mathbf{T[1, x] = x}$

In[3]:= $\mathbf{T[n\_, x\_] := T[n, x] = 2xT[n-1, x] - T[n-2, x]}$

Для просмотра многочлена Чебышева, например, для $n = 7$ необходимо выполнить команду:

In[4]:= $\mathbf{T[7, x]//FullSimplify//Expand}$

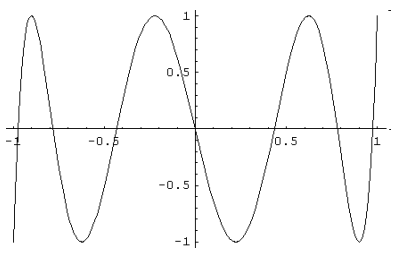
Out[4]= $-7x + 56x^3 - 112x^5 + 64x^7$

Старший коэффициент многочлена Чебышева $\mathbf{T[n, x]}$ равен $\mathbf{2^{n-1}}$, поэтому приведенный многочлен Чебышева имеет вид: $\frac{1}{2^{n-1}}\mathbf{T[n, x]}$.

График многочлена Чебышева принадлежит прямоугольнику $[-1,1] \times [-1,1]$.

In[5]:= $\mathbf{Plot[Evaluate[T[7, x]], \{x, -1, 1\}]}$

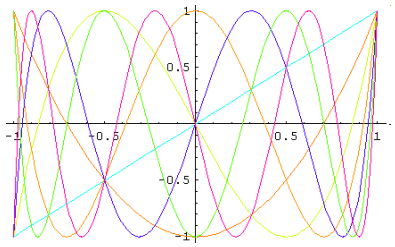
Out[5]=



Построим графики нескольких многочленов Чебышева:

In[6]:=
```
color = Table[{Hue[Random[]]}, {i, 1, 7}]
Plot[Evaluate[Table[T[i, x], {i, 1, 7}]], {x, -1, 1},
PlotStyle → color]
```

Out[6]=



Рассмотрим пример. Возьмем функцию

In[1]:=
```
f[x_] = Sin[x];
```

как в примере предыдущего пункта, и построим для нее многочлен Лагранжа 5-го порядка и наилучшего приближения на отрезке $[-6, 6]$, применяя многочлены Чебышева. Построим графики.

Для этого введем неравноостоящие узлы x_i, $i = 0, 1, ..., 5$, по формуле (6), где t_i есть корни (4) многочлена Чебышева 6-го порядка на отрезке $[-1, 1]$; $a = -6$, $b = 6$.

In[2]:=
```
a = -6; b = 6; n = 6;
```

In[3]:=
```
Table[t_i = Cos[(2i+1)π/2n], {i, 0, n - 1}]//N
Table[x_i = (a+b)/2 + (a-b)/2 t_i, {i, 0, n - 1}]//N
```

Out[3]= {0.965926, 0.707107, 0.258819, -0.258819, -0.707107, -0.965926}

Out[4]= {-5.79555, -4.24264, -1.55291, 1.55291, 4.24264, 5.79555}

В результате получим таблично заданную функцию

In[5]:=
```
fs = Table[{x_i, f[x_i]}, {i, 0, n - 1}]//N
```

Out[5]=

{{-5.79555, 0.468534}, {-4.24264, 0.891682}, {-1.55291, -0.99984},
{ 1.55291, 0.99984}, {4.24264, -0.891682}, {5.79555, -0.468534}}

Для табличной функции **fs** построим многочлен Лагранжа 5-го порядка. Этот многочлен и будет наилучшим приближением для данной функции $f(x)$ среди всех ее аппроксимационных многочленов Лагранжа 5-го порядка.

Построим график функции $f(x)$:

In[6]:=
```
gr1 = Plot[f[x], {x, -6, 6}, PlotStyle → {Hue[0.3]}]
```

Out[6]=

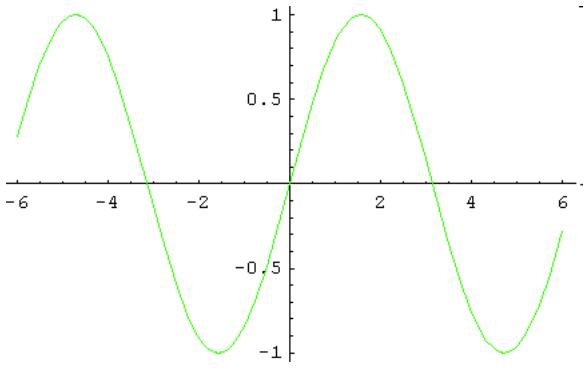
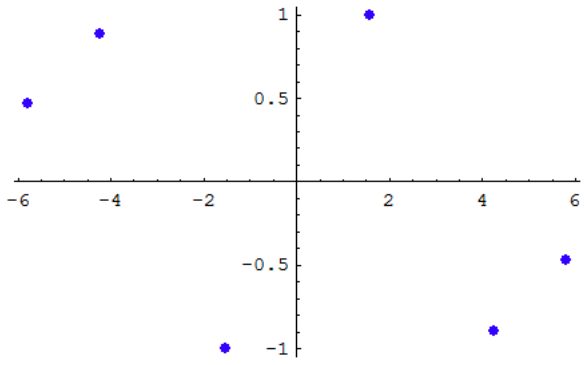


график таблично заданной функции

In[7]:= **gr2 = ListPlot[fs, PlotStyle → {Hue[0.7], PointSize[0.02]}]**

Out[7]=



Построим многочлен Лагранжа:

In[8]:=

$$\mathbf{L_{n-1}[x\_] = \sum_{j=0}^{n-1} f[x_j] \prod_{i=0}^{n-1} If[i == j, 1, \frac{x - x_i}{x_j - x_i}]//N//Expand//Chop}$$

Out[8]=

$0.86379x - 0.09608x^3 + 0.002023x^5$

Найдем оценку (7) для построенного многочлена. Так как $n = 6$ есть степень многочлена Чебышева, то в (7) надо сделать замену $n \to n - 1$:

$$|f(x) - L(x)| \leqslant \frac{M_{n-1}}{n!} \frac{(b-a)^n}{2^{2n-1}} \tag{8}$$

Модуль любой производной синуса не больше 1, следовательно $|M_5| \leqslant 1$. Поэтому правая часть (8) оценивается

In[8]:= $\frac{1}{n!} \frac{(b-a)^n}{2^{2n-1}}$

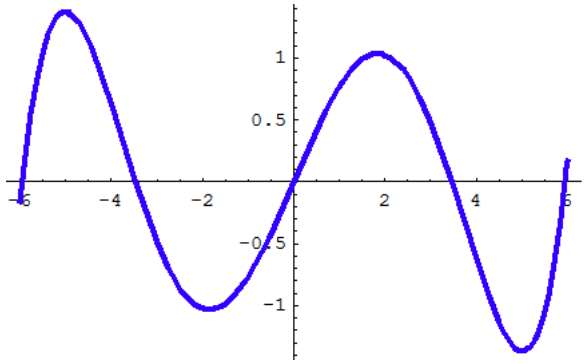
Out[8]= 2.025

Сравните с оценкой аппроксимации функции $sin(x)$ многочленом Лагранжа, построенного по табличной функции с равноотстоящими узлами, в п.1.

Построим график многочлена Лагранжа:

In[9]:= **gr3 = Plot[L$_{n-1}$[x], {x, −6, 6}, PlotStyle → {Hue[0.7]}]**

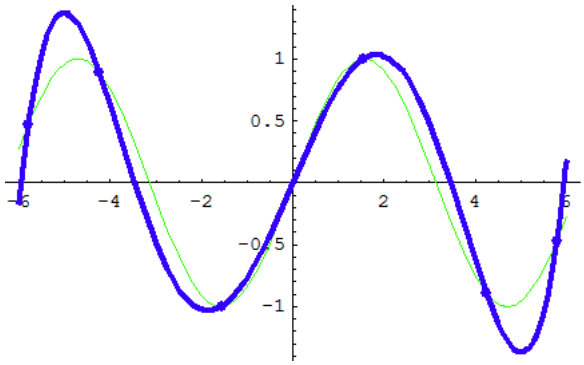
Out[9]=



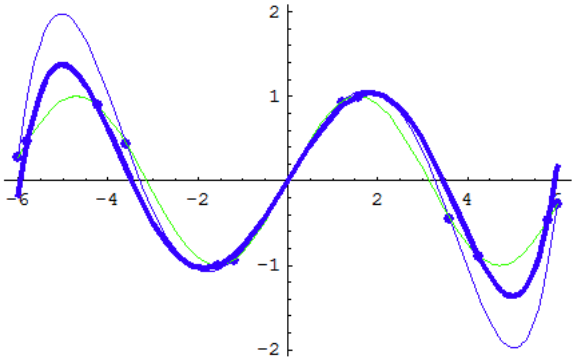
Теперь совместим три графика для сравнения:

In[10]:= **Show[gr1, gr2, gr3]**

Out[10]=



Для сравнения приведем графики функции $f(x)$, и многочленов Лагранжа построенных в этом и предыдущем пунктах.



n.3 Многочлен Ньютона

Дана табличная функция (x_i, y_i), $i = 0, 1, ..., n$, с равноотстоящими узлами, следующими с шагом h.

Интерполяционным многочленом Ньютона называется многочлен n-ой степени $P_n(x)$, удовлетворяющий равенствам

$$P_n(x_i) = y_i, \tag{1}$$

$i = 0, 1, ..., n$, и имеющий следующий вид:

$$P_n(x) = y_0 + A_1(x - x_0) + A_2(x - x_0)(x - x_1) + ...$$

$$+A_n(x - x_0)(x - x_1)...(x - x_{n-1}), \tag{2}$$

где A_i - постоянные.

Из определения многочлена Лагранжа следует, что многочлен Ньютона есть и многочлен Лагранжа. Вид многочлена Ньютона позволяет не пересчитывать все коэффициенты многочлена при добавлении еще одного узла справа - для получения

нового многочлена Ньютона, построенного на n+1 узле, достаточно добавить к многочлену Ньютона, построенному на n узлах, только одно последнее слагаемое.

Найдем коэффициенты A_i. Запишем условия (1) для многочлена (2). При i=0 соотношение (1) для многочлена (2) выполняется: $P_n(x_0) = y_0$. Пусть i=1.

$$P_n(x_1) = y_0 + A_1(x_1 - x_0) = y_1, \quad \text{отсюда} \quad A_1 = \frac{y_1 - y_0}{h}.$$

Обозначим $\triangle_1 y_0 = y_1 - y_0$, тогда $A_1 = \frac{\triangle_1 y_0}{h}$.

Здесь удобно ввести следующие определение: назовем конечными разностями 0-го порядка величины вида $\triangle_0 y_k = y_k$, $k = 0, 1, 2, ..., n$, а конечными разностями m-го порядка величины вида

$$\triangle_m y_k = \triangle_{m-1} y_{k+1} - \triangle_{m-1} y_k,$$

$m = 0, 1, 2, ...n$, $k = 0, 1, 2, ..., n - m$. В таблице ниже $n = 10$.

Вычисление таких величин легко провести по следующей таблице конечных разностей:

x_i	y_i	$\triangle_1 y_i$	$\triangle_2 y_i$	$\triangle_3 y_i$	$\triangle_4 y_i$	$\triangle_5 y_i$	$\triangle_6 y_i$	$\triangle_7 y_i$	$\triangle_8 y_i$	$\triangle_9 y_i$	$\triangle_{10} y_i$
x_0	y_0	$\triangle_1 y_0$	$\triangle_2 y_0$	$\triangle_3 y_0$	$\triangle_4 y_0$	$\triangle_5 y_0$	$\triangle_6 y_0$	$\triangle_7 y_0$	$\triangle_8 y_0$	$\triangle_9 y_0$	$\triangle_{10} y_0$
x_1	y_1	$\triangle_1 y_1$	$\triangle_2 y_1$	$\triangle_3 y_1$	$\triangle_4 y_1$	$\triangle_5 y_1$	$\triangle_6 y_1$	$\triangle_7 y_1$	$\triangle_8 y_1$	$\triangle_9 y_1$	
x_2	y_2	$\triangle_1 y_2$	$\triangle_2 y_2$	$\triangle_3 y_2$	$\triangle_4 y_2$	$\triangle_5 y_2$	$\triangle_6 y_2$	$\triangle_7 y_2$	$\triangle_8 y_2$		
x_3	y_3	$\triangle_1 y_3$	$\triangle_2 y_3$	$\triangle_3 y_3$	$\triangle_4 y_3$	$\triangle_5 y_3$	$\triangle_6 y_3$	$\triangle_7 y_0$			
x_4	y_4	$\triangle_1 y_4$	$\triangle_2 y_4$	$\triangle_3 y_4$	$\triangle_4 y_4$	$\triangle_5 y_4$	$\triangle_6 y_4$				
x_5	y_5	$\triangle_1 y_5$	$\triangle_2 y_5$	$\triangle_3 y_5$	$\triangle_4 y_5$	$\triangle_5 y_5$					
x_6	y_6	$\triangle_1 y_6$	$\triangle_2 y_6$	$\triangle_3 y_6$	$\triangle_4 y_6$						
x_7	y_7	$\triangle_1 y_7$	$\triangle_2 y_7$	$\triangle_3 y_7$							
x_8	y_8	$\triangle_1 y_8$	$\triangle_2 y_8$								
x_9	y_9	$\triangle_1 y_9$									
x_{10}	y_{10}										

Так, например, $\triangle_1 y_2$ получается вычитанием ближайших элементов предыдущего столбца $y_3 - y_2$, а, например, $\triangle_3 y_4$ есть разность $\triangle_2 y_5 - \triangle_2 y_4$ и так далее.

Для i=2 соотношение (1) запишется в следующем виде:

$$P_n(x_2) = y_0 + A_1(x_2 - x_0) + A_2(x_2 - x_0)(x_2 - x_1) = y_2$$

или

$$y_0 + \frac{\triangle_1 y_0}{h} 2h + A_2 2h^2 = y_2$$

Отсюда получаем

$$A_2 = \frac{y_2 - y_0 - 2\ \triangle_1 y_0}{2h^2} \tag{3}$$

Преобразуем разность $y_2 - y_0 - 2\ \triangle_1 y_0 = y_2 - y_0 - 2(y_1 - y_0) = y_2 - y_1 - (y_1 - y_0) = \triangle_1 y_1 - \triangle_1 y_0$. Отсюда и из (3):

$$A_2 = \frac{\triangle_1 y_1 - \triangle_1 y_0}{2h^2},$$

или

$$A_2 = \frac{\triangle_2 y_0}{2h^2}$$

Применяя полную индукцию, можно показать, что

$$A_n = \frac{\triangle_n y_0}{n!h^n}.$$

Подставляя найденные коэффициенты в (2) получим многочлен

$$P_n(x) = y_0 + \frac{\triangle_1 y_0}{1!h}(x - x_0) + \frac{\triangle_2 y_0}{2!h^2}(x - x_0)(x - x_1) + ...+$$

$$\frac{\triangle_n y_0}{n!h^n}(x - x_0)(x - x_1)...(x - x_{n-1}), \tag{4}$$

который называется *первым интерполяционным многочленом Ньютона.* Он позволяет интерполировать значения табличной функции в окрестности базового узла x_0. Для этого введем новую переменную $q = (x - x_0)/h$. Отсюда найдем $x = x_0 + qh$, тогда $x - x_i = x_0 + qh - x_0 - ih = h(q - i)$. Подставим в (4) и получим *первую интерполяционную формулу Ньютона*:

$$P_n(x) = P_n(x_0 + qh) = y_0 + q\frac{\triangle_1 y_0}{1!} + \frac{q(q-1)}{2!}\ \triangle_2 y_0 + ...+$$

$$\frac{q(q-1)...(q-n+1)}{n!}\ \triangle_n y_0$$

Значение $P_n(x)$ объявляем значением исходной табличной функции в точке x. Формула Ньтона обычно применяется для интерполирования в точки $x \in (x_0, x_1)$ и $x < x_0$.

Многочлен (4) можно записать компактно так

$$P_n(x) = \sum_{k=0}^{n}(\frac{\triangle_k y_0}{k!h^k}\prod_{i=0}^{k-1}(x - x_i)), \tag{5}$$

считая, что $\prod_{i=0}^{-1}(x - x_i) = 1$.

Аналогично можно получить *второй интерполяционный многочлен Ньютона*, для которого базовой точкой будет конец отрезка x_n, а коффициенты взяты с нижней стороны треугольника коэффициентов:

$$P_n(x) = y_n + \frac{\triangle_1 y_{n-1}}{1!h}(x - x_n) + \frac{\triangle_2 y_{n-2}}{2!h^2}(x - x_n)(x - x_{n-1}) + ...+$$

$$\frac{\triangle_{n,0}}{n!h^n}(x - x_n)(x - x_{n-1})...(x - x_1).$$

или в компактной записи:

$$P_n(x) = \sum_{k=0}^{n}(\frac{\triangle_k y_{n-k}}{k!h^k}\prod_{i=0}^{k-1}(x - x_{n-i})), \tag{5}$$

считая, что $\prod_{i=0}^{-1}(x - x_i) = 1$.

Вторая интерполяционная формула Ньютона:

$$P_n(x) = P_n(x_n + qh) = y_n + \frac{q}{1!}\,\triangle_{n-1}y_1 + \frac{q(q-1)}{2!}\,\triangle_{n-2}y_2 + ...+$$

$$\frac{q(q+1)...(q+n-1)}{n!}\,\triangle_0 y_n$$

Построим многочлен Ньютона для табличной функции, построенной с помощью функции

In[1]:= $\mathbf{f[x\_] = Sin[x^2 + 6]}$;

Out[1]= $sin(x^2 + 6)$

Рассмотрим отрезок [-6,18], число узлов возьмем равным n+1, где

In[2]:= $\mathbf{n = 10}$;

Шаг следования узлов равен

In[3]:= $\mathbf{h = \frac{18-(-6)}{10} = 2.4}$;

Будем округлять данные до второго знака после запятой. Округление позволит написать следующую таблицу в более компактном виде. Для этого введем функцию округления

In[4]:= $\mathbf{rn[x\_, n\_] := Round[x10^n]/10^n//N}$

Введем координаты узлов

In[5]:= $\mathbf{Table[x_i = -6 + h\ i, \{i, 0, n\}]}$

Out[5]= {-6, -3.6, -1.2, 1.2, 3.6, 6., 8.4, 10.8, 13.2, 15.6, 18.}

и введем табличную функцию

In[6]:= $\mathbf{fs = Table[\{x_i, y_i = f[x_i]\}, \{i, 0, n\}]//N}$

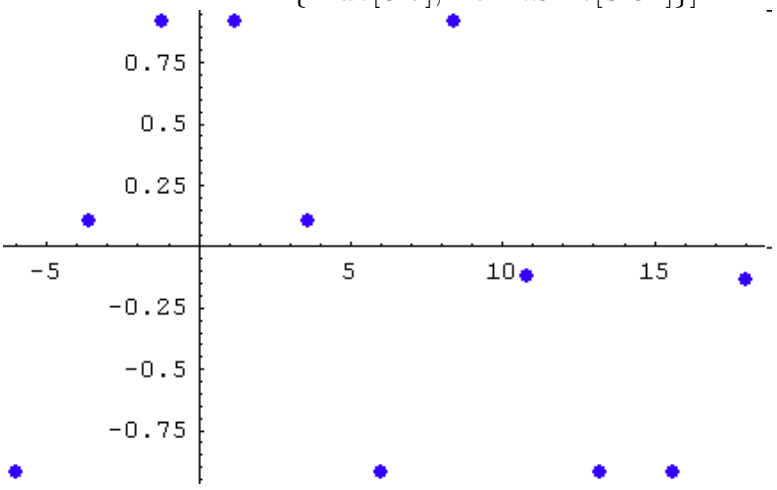
Округлим данные

In[7]:= $\mathbf{fs = rn[fs, 2]}$

Out[7]= {{-6., -0.92}, {-3.6, 0.11}, {-1.2, 0.92}, {1.2, 0.92}, {3.6, 0.11},
{ 6., -0.92}, {8.4, 0.92}, { 10.8, -0.12}, {13.2, -0.92}, {15.6, -0.92},
{18., -0.13}}

Построим график табличной функции:

In[8]:= $\mathbf{grpoint = ListPlot[fs, PlotStyle \to \{Hue[0.7], PointSize[0.02]\}]}$



In[9]:= $\mathbf{rn[Table[\triangle_0 y_k = y_k, \{k, 0, n\}], 2]}$

Out[9]= {-0.92, 0.11, 0.92, 0.92, 0.11, -0.92, 0.92, -0.12, -0.92, -0.92, -0.13}

Результат команды выведем в виде следующей таблицы. Подчеркнутые элементы определяют коэффициенты A_n.

In[9]:=
```
rn[Table[△m yk =△m−1 yk+1 − △m−1 yk,
  {m, 1, n}, {k, 0, n − m}], 2]//Chop
```

x_i	y_i	$\triangle_1 y_i$	$\triangle_2 y_i$	$\triangle_3 y_i$	$\triangle_4 y_i$	$\triangle_5 y_i$	$\triangle_6 y_i$	$\triangle_7 y_i$	$\triangle_8 y_i$	$\triangle_9 y_i$	$\triangle_{10} y_i$
−6	<u>-0.92</u>	<u>1.03</u>	<u>-0.22</u>	<u>-0.58</u>	<u>0.58</u>	<u>0</u>	<u>1.91</u>	<u>-15.14</u>	<u>57.32</u>	<u>-157.46</u>	<u>357.89</u>
−3.6	0.11	0.81	-.81	0	0.58	1.91	-13.22	42.18	-100.14	200.43	
−1.2	0.92	0	-0.81	0.58	2.5	-11.31	28.95	-57.96	100.29		
1.2	0.92	-0.81	-0.22	3.08	-8.81	17.64	-29.01	42.33			
3.6	0.11	-1.03	2.86	-5.73	8.83	-11.36	13.32				
6	-0.92	1.84	-2.87	3.1	-2.53	1.96					
8.4	0.92	-1.04	0.23	0.57	-0.58						
10.8	-0.12	-0.8	0.8	-0.01							
13.2	-0.92	0	0.79								
15.6	-0.92	0.79									
18	-0.13										

Построим многочлен Ньютона по формуле (5):

In[9]:=

$$(\mathbf{g[x\_]} = \sum_{\mathbf{k=0}}^{\mathbf{n}} (\frac{\triangle_{\mathbf{k}} \mathbf{y_0}}{\mathbf{h^k k!}} \prod_{\mathbf{i=0}}^{\mathbf{k-1}} (\mathbf{x - x_i}))//\mathbf{N}//\mathbf{Chop}$$

Out[9]=

$$-0.92 + 0.43(6 + x) - 0.02(3.6 + x)(6. + x) - 0.01(1.2 + x)(3.6 + x)(6. + x) + 0.001(-1.2 + x)\ (1.2 + x)\ (3.6\ + x)\ (6. + x) + 0.000014(-6. + x)\ (-3.6 + x)\ (-1.2 + x)\ (1.2 + x)\ (3.6 + x)\ (6.\ + x) - 6.55\ 10^{-6}\ (-8.4 + x)\ (-6. + x)\ (-3.6 + x)\ (-1.2 + x)\ (\ 1.2 + x)\ (3.6\ + x)\ (6. + x) + \ 1.3\ 10^{-6}\ (-10.8 + x)\ (-8.4 + x)\ (-6. + x)\ (-3.6 + x)\ (-1.2 + x)\ (1.2 + x)\ (3.6 + x)\ (6. + x) - 1.64\ 10^{-7}\ (-13.2 + x)\ (-10.8 + x)\ (-8.4 + x)\ (-6. + x)\ (-3.6 + x)\ (-1.2 + x)\ (1.2 + x)\ (3.6\ + x)\ (6. + x) + 1.56\ 10^{-8}\ (-15.6 + x)\ (-13.2 + x)\ (-10.8 + x)\ (-8.4 + x)\ (-6. + x)\ (-3.6 + x)\ (-1.2 + x)\ (1.2 + x)\ (3.6\ + x)\ (6. + x)$$

При упрощении получим

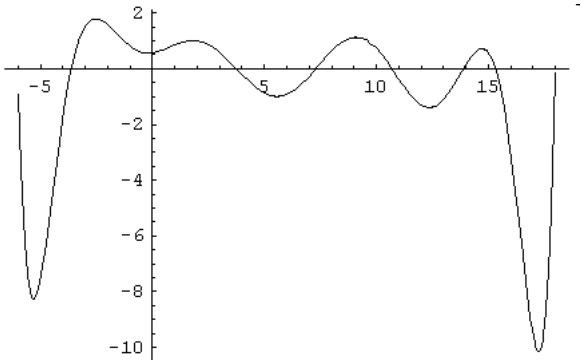
In[10]:=
```
%//FullSimplify//Expand
```

Out[10]=

$$0.57747 + 0.12766x + 0.26785x^2 - 0.10143x^3 - 0.02256x^4 + 0.00908x^5 - 0.00031x^6 - 0.00014x^7 + 0.00002x^8 - 9.10893\ 10^{-7}x^9 + 1.5555\ 10^{-8}x^{10}$$

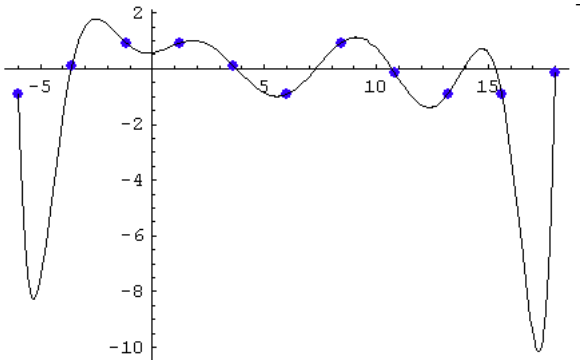
Построим график многочлена Ньютона

In[9]:=
```
gr = Plot[g[x], {x, −6, 18}]
```



Совместим графики многочлена Ньютона и график табличной функции

In[9]:= **Show**[**grpoint**, **gr**]



n.4 Кубические сплайны

Рассмотрим аппроксимацию функций кубическими сплайнами.

Дадим определение кубического сплайна на отрезке $[p, q]$. Разобьем отрезок $[p, q]$ на n частей точками (узлами) $p = x_0 < x_1 < ... < x_n = q$ и на каждом отрезке $[x_i, x_{i+1}]$, $i = 0, 1, ..., n-1$, рассмотрим кубический многочлен $s_i(x) = a_i\ x^3 + b_i\ x^2 + c_i\ x + d_i$ такой, что

$$s_i(x_i) = f_i, \quad s_i(x_{i+1}) = f_{i+1}, \quad s_i'(x_i) = m_i \quad s_i'(x_{i+1}) = m_{i+1} \tag{1}$$

при некоторых табличных функциях (x_i, f_i), (x_i, m_i), $i = 0, 1, ..., n-1$. Условия (1) означают, что кубические многочлены и касательные к кубическим многочленам в узлах принимают заданные значения.

Функция $spl(x)$, определенная на $[p, q]$ и такая, что ее сужение на каждом отрезке $[x_i, x_{i+1}]$ совпадает с $s_i(x)$, называется *кубическим сплайном*. Величины m_i называются наклонами сплайна в узлах.

Разность $3 - k$, где k - порядок наивысшей непрерывной производной сплайна $spl(x)$, называется дефектом сплайна.

Кубический сплайн, удовлетворяющий соотношениям (1), будет, по крайней мере, один раз непрерывно дифференцируемым, то есть будет сплайном дефекта (2). Для

построения сплайна дефекта 1 потребуем равенство вторых производных кубических парабол:

$$s''_{i-1}(x_i) = s''_i(x_i) \tag{2}$$

для $i = 1, ..., n-1$. Эти равенства задают $n-1$ линейное соотношение на наклоны сплайна во внутренних узлах.

Построим в программе Mathematica кубический сплайн по данным табличным функциям (x_i, f_i), (x_i, m_i), $i = 0, 1, ..., n-1$.

Пусть $s(x) = a\,x^3 + b\,x^2 + c\,x + d$ кубический многочлен на отрезке $[x_i, x_{i+1}]$. Тогда условия (1) можно записать в виде системы уравнений

```
In[1]:= eq = {a x_i^3 + b x_i^2 + c x_i + d == f_i,
              a x_(i+1)^3 + b x_(i+1)^2 + c x_(i+1) + d == f_(i+1),
              3 a x_i^2 + 2b x_i + c == m_i,
              3 a x_(i+1)^2 + 2b x_(i+1) + c == m_(i+1)}
```

Решаем эту систему

```
In[2]:= sq = FullSimplify[Solve[eq, {a, b, c, d}]]
```

получим значения коэффициентов

Out[2]=

$$\{\{d \to \frac{f_{i+1}(x_i - 3\,x_{i+1})\,x_i^2 + x_{i+1}\,(-m_{i+1}\,x_i^3 + (3\,f_i + (m_{i+1} - m_i)\,x_i)\,x_{i+1}\,x_i - (f_i - m_i\,x_i)\,x_{i+1}^2)}{(x_i - x_{i+1})^3},$$
$$a \to \frac{-2\,f_i + 2\,f_{i+1} + (m_i + m_{i+1})\,(x_i - x_{i+1})}{(x_i - x_{i+1})^3},$$
$$b \to \frac{3\,f_i\,(x_i + x_{i+1}) - 3\,f_{i+1}(x_i + x_{i+1}) - (x_i - x_{i+1})((m_i + 2\,m_{i+1})\,x_i + (2\,m_i + m_{i+1})x_{i+1}))}{(x_i - x_{i+1})^3},$$
$$c \to \frac{m_{i+1}\,x_i^3 - (m_i + 2\,m_{i+1})x_{i+1}^2\,x_i + (-6\,f_i + 6\,f_{i+1} + (2\,m_i + m_{i+1})\,x_i)\,x_{i+1}\,x_i - m_i\,x_{i+1}^3)}{(x_i - x_{i+1})^3}\}\}$$

Подставим найденные коэффициенты в функцию s(w) и проведем упрощение результата. Здесь заменили аргумент на **w**, чтобы не было ненужных подстановок. Получим:

```
In[3]:= s[w_] = FullSimplify[a w^3 + b w^2 + c w + d /. sq][[1]]
Out[3]= 1/(x_i - x_(i+1))^3 ((w - x_i) (f_(i+1) (w - x_i) (2 w + x_i - 3 x_(i+1)) + (m_(i+1) (w - x_i) +
        m_i (w - x_(i+1))) (w - x_(i+1)) (x_i - x_(i+1))) - f_i (w - x_(i+1))^2 (2 w - 3 x_i +
        x_(i+1)))
```

Объединим все кубические многочлены средствами программы и создадим команду с именем $spl(x)$, которая возвращает для каждого значения x значение соответствующего кубического многочлена в данном x. Сначала по заданному x высчитывается индекс i, затем вычисляется значение функции $s(x)$.

```
In[4]:= spl[x_] := Block[{i}, i = IntegerPart[(x - a)/h]; s[x]]
```

Предположим теперь, что функции $f(x)$ дважды непрерывно дифференцируема на отрезке $[a, b]$. Кубический сплайн $spl(x)$ аппроксимирует функцию $f(x)$ с

а) дефектом 2, если положить $f_i = f(x_i)$ и $m_i = f'(x_i)$ для все $i = 0, 1, ..., n$;

б) дефектом 1, если положить $f_i = f(x_i)$ для все $i = 0, 1, ..., n$ и $s''_{i-1}(x_i) = s''_i(x_i)$ для все внутренних узлов $i = 1, ..., n-1$. При этом, на граничных узлах можно положить, например, $m_0 = f'(x_0)$, $m_n = f'(x_n)$.

Данные требования позволяют определить табличные функции (x_i, f_i), (x_i, m_i), $i = 0, 1, ..., n-1$, по которым строится сплайн $spl[x]$. В первом случае сплайн будет, по крайней мере, один раз непрерывно дифференцируемым, во втором - непрерывной будет и вторая производная сплайна.

Теорема. Если $f \in C^{k+1}[a,b]$, $0 \leqslant k \leqslant 2$, то интерполяционный сплайн $spl(x)$ с наклонами, заданными способами а), б), удовлетворяют неравенству

$$\max_{[x_i,x_{i+1}]} |f^{(m)}(x) - spl^{(m)}(x)| \leqslant c\, h^{k+1-m} \max_{[a,b]} |f^{(k+1)}(x)|,$$

где $i = 0, 1, ..., n-1$, $m = 0, 1, 2$, а c - не зависящая от h, i, f постоянная.

Теперь рассмотрим конкретную задачу. Функцию

In[5]:= **f[x_] = x^2 Sin[x/2]**

определенную на отрезке [1,9], аппроксимируем кубическим сплайном. Построим графики функции и сплайна.

Пусть шаг следования узлов $h = 1$. Вводим начальную точку отрезка и шаг

In[6]:= **a = 1;**
h = 1;

Сначала построим кубический сплайн дефекта 2.

Определим начальные условия сплайна, то есть условия (1).

In[7]:= **Table[{x_i = a + i h, f_i = f[x_i], m_i = f'[x_i]}, {i, 0, 8}]//N**

Out[7]= {{ 1 , 0.4794, 4.1929}, {2, 3.3659, 13.3394}, { 3, 8.9774, 18.9099},
{ 4, 14.5487, 11.8356},{ 6, 5.0803, -48.3792}, { 5, 14.9618, -12.0887},
{ 7, -17.1883, -83.5624}, { 8, -48.4353, -99.0763},
{ 9, -79.1799, -78.3983}},

Посмотрим как выглядит график сплайна на чертеже. Для этого нарисуем отдельно кубические параболы:

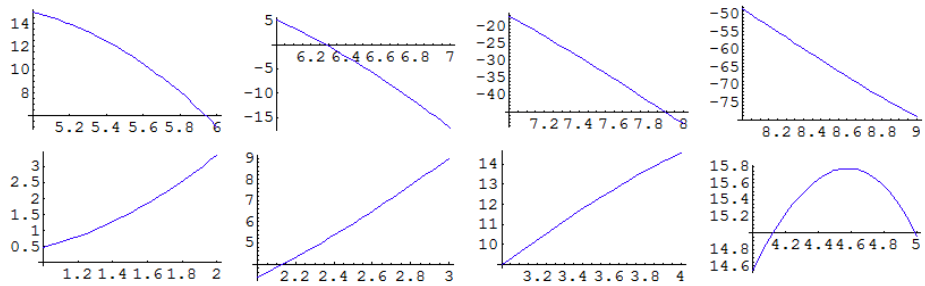
In[8]:= **gr = Table[**
Plot[s[w], {w, x_i, x_{i+1}}, PlotStyle → {Hue[0.7]}],
{i, 0, 3}];

In[9]:= **gr1 = Table[**
Plot[s[w], {w, x_i, x_{i+1}}, PlotStyle → {Hue[0.7]}],
{i, 4, 7}];

Для экономии места воспользовались функцией **GraphicsArray** и построим отдельно все кубические параболы:

In[10]:= **Show[GraphicsArray[gr1, gr]]**

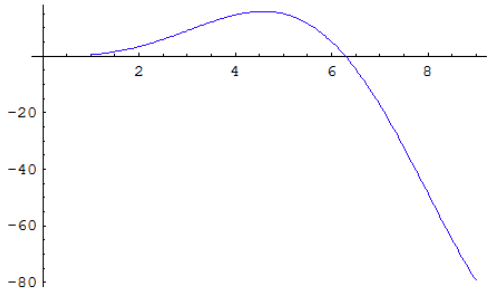
Out[10]=



Строим график сплайна - выведем на одном чертеже все кубические параболы. Можно, конечно, построить график функции $spl[x]$, что будет одно и тоже.

In[11]:= **Show[gr, gr1]**

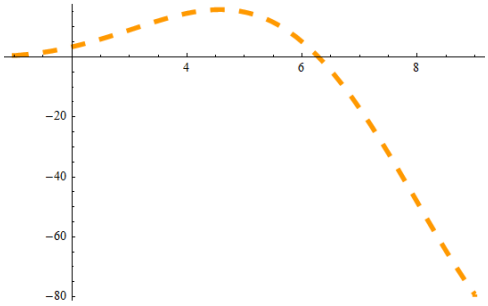
Out[11]=



Строим график данной функции для сравнения с графиком сплайна:

In[12]:= **gr2 = Plot[f[x], {x, 1, 9}, PlotStyle →**
{Hue[0.1], Dashing[{0.03}], Thickness[0.01]}]

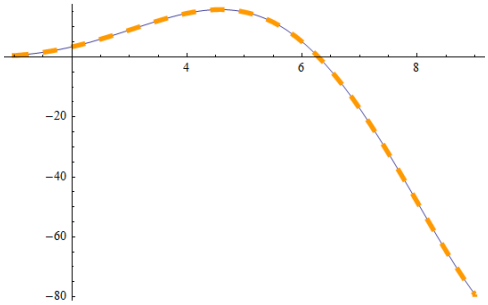
Out[12]=



Совмещаем графики

In[13]:= **Show[gr, gr1, gr2]**

Out[13]=



Построим теперь кубический сплайн дефекта 1.

Считаем, что первые 6 команд уже выполнены. Очистим значения параметров m_i:

In[7]:= **Remove[m];**

Зададим точки разбиения отрезка и значения табличной функции (x_i, f_i):

In[8]:= **Table[{x$_i$ = a + (i − 1)h, f$_i$ = f[x$_i$]//N}, {i, 1, 9}];**

В функции $s[w]$ сделаем замену $i \to i+1$ и получим смежную кубическую параболу $s1[w]$:

In[9]:= **s1[w_] = s[w]/.i → i − 1**

Потребуем равенство вторых производных у смежных парабол во внутренних узлах:

In[10]:= **eq1 = Table[s''[x$_i$] == s1''[x$_i$], {i, 2, 8}]//FullSimplify**

Out[10]= $\{2m_1 + 8.m_2 + 2.m_3 = 50.9882,$
$2m_2 + 8.m_3 + 2.m_4 = 67.0972,$
$2m_3 + 8.m_4 + 2.m_5 = 35.9061,$
$1.m_4 + 4.m_5 + 1.m_6 + 28.4053 = 0,$
$2m_5 + 8.m_6 + 2.m_7 + 192.901 = 0,$
$2m_6 + 8.m_7 + 2.m_8 + 321.094 = 0,$
$2m_7 + 8.m_8 + 2.m_9 + 371.949 = 0\}$

Добавим к этому списку еще два уравнения:

In[11]:= **eq2 = {m$_1$ == f'[1], m$_9$ == f'[9]}//N**

$\{m_1 = 1.39764, m_9 = -26.1328\}$

и получим следующую систему уравнений:

In[12]:= **eq = Union[eq1, eq2]//N**

Out[12]= $\{m_1 = 1.39764,$
$2m_1 + 8.m_2 + 2.m_3 = 50.9882,$
$2m_2 + 8.m_3 + 2.m_4 = 67.0972,$
$2m_3 + 8.m_4 + 2.m_5 = 35.9061,$
$1.m_4 + 4.m_5 + 1.m_6 + 28.4053 = 0,$
$2m_5 + 8.m_6 + 2.m_7 + 192.901 = 0,$
$2m_6 + 8.m_7 + 2.m_8 + 321.094 = 0,$
$2m_7 + 8.m_8 + 2.m_9 + 371.949 = 0,$
$m_9 = -26.1328\}$

Введем неизвестные системы уравнений

In[13]:= **per = Table[m$_i$, {i, 1, 9}]**

Out[13]= $\{m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8, m_9\}$

и разрешим эту линейную систему с помощью встроенной команды

In[14]:= **ss = Solve[eq, per][[1]]**

Out[14]= $\{m_1 \to 1.39764, m_2 \to 4.45097, m_3 \to 6.29258, m_4 \to 3.92734,$
$m_5 \to -4.04888, m_6 \to -16.1371, m_7 \to -27.8532,$
$m_8 \to -32.9972, m_9 \to -26.1328\}$

Таким образом, получены следующие наклоны сплайна дефекта 1:

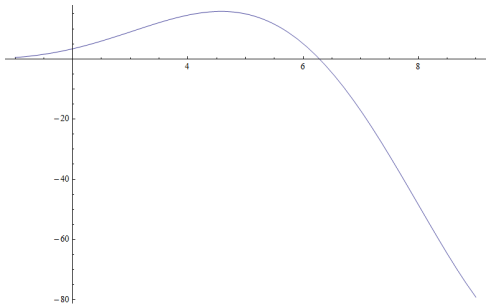
In[15]:= **Table[m$_i$ = ss[[i, 2]], {i, 1, 9}]**

Out[15]= {1.39764, 4.45097, 6.29258, 3.92734, -4.04888, -16.1371,
-27.8532, -32.9972, -26.1328}

Построим график сплайна. В вызванную функцию $spl(x)$ будут подставлены найденные наклоны сплайна.

In[16]:= **gr5 = Plot[spl[x], {x, 1, 9}, PlotStyle →**
{Hue[0.1], Dashing[{0.03}], Thickness[0.01]}]

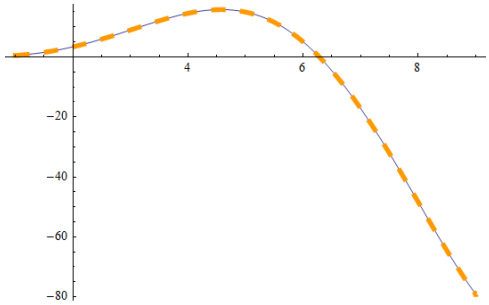
Out[16]=



Совместим графики сплайна и функции.

In[17]:= **Show[gr2, gr5]**

Out[17]=



n.5 Метод наименьших квадратов

Метод наименьших квадратов позволяет построить интерполяционную функцию для данной табличной функции или аппроксимационную функцию для данной измеримой функции, наиболее близкую функцию в определенном ниже смысле, среди всех функций, принадлежащих данному классу функций. Если табличная функция f есть сужение на множестве узлов x_0, x_1, ..., x_m некоторой функции $f(x)$, определенной на отрезке $[x_0, x_m]$, то метод наименьших квадратов позволяет построить наиболее близкую аппроксимацию функции $f(x)$ функцией из данного класса функций. При этом, построенную аппроксимационную функцию, объявляют самой близкой по методу наименьших квадратов и к табличной функции f, не смотря на то, что эти функции определены на разных множествах.

Для измеримых функций $f = f(x)$ $g = g(x)$, определенных на отрезке $[a, b]$, определим скалярное произведение равенством

$$(f, g) = \int_a^b f(x)g(x)dx. \tag{1}$$

Для табличных функций $f = \{(x_i, f_i)\}$, $g = \{(x_i, g_i)\}$, $i = 0, 1, ..., m$, определенных над одним и тем же множеством узлов $\{x_0, ..., x_n\}$, определим скалярное произведение равенством

$$(f, g) = \sum_{i=0}^{m} f_i g_i. \tag{2}$$

Расстояние между функциями f и g в обоих случаях определяется равенством

$$\rho(f, g) = \sqrt{(f - g, f - g)}. \tag{3}$$

Функция

$$\Phi(x) = c_0\varphi_0(x) + c_1\varphi_1(x) + ... + c_n\varphi_n(x), \tag{4}$$

определенная на отрезке $[a, b]$, называется *обобщенным многочленом* по системе функция $\varphi_0(x), \varphi_1(x), ..., \varphi_n(x)$, коэффициенты $c_0, ..., c_n$ - константы, число n называется порядком многочлена.

Известно, что функции $\varphi_0(x), \varphi_1(x), ..., \varphi_n(x)$ линейно независимы, тогда и только тогда, когда определитель Грама этой системы не равен нулю:

$$\begin{vmatrix} (\varphi_0,\varphi_0) & (\varphi_0,\varphi_1) & ... & (\varphi_0,\varphi_n) \\ (\varphi_1,\varphi_0) & (\varphi_1,\varphi_1) & ... & (\varphi_1,\varphi_n) \\ . & . & . & . \\ (\varphi_n,\varphi_0) & (\varphi_n,\varphi_1) & ... & (\varphi_n,\varphi_n) \end{vmatrix} \neq 0.$$

Например, функции $\varphi_0(x) = 1$, $\varphi_1(x) = x, ... \ \varphi_n(x) = x^n$ линейно независимы на всей числовой прямой. Обобщенный многочлен по такой системе функций есть обычный многочлен. Функции $\varphi_0(x) = x$, $\varphi_1(x) = \frac{1}{x}$ линейно независимы на отрезке $[1, 2]$. Действительно, равенство $c_0 x + c_1 \frac{1}{x} = 0$ при всех $x \in [1, 2]$ возможно лишь при $c_0 = c_1 = 0$.

Непрерывный случай. Пусть дана измеримая функция $f = f(x)$, определенная на отрезке $[a, b]$. Рассмотрим класс J обобщенных многочленов порядка n по фиксированной системе функции $\varphi_0(x), \varphi_1(x), ..., \varphi_n(x)$. Определим обобщенный многочлен из класса J наилучшего среднеквадратичного приближения для данной измеримой функции $f = f(x)$ на отрезке $[a, b]$, то есть, найдем такой обобщенный многочлен $\Phi_0 = \Phi_0(x) \in J$, что

$$\rho(f, \Phi_0) = \min_{\Phi = \Phi(x) \in J} \rho(f, \Phi),$$

где расстояние ρ вычисляется по формуле (3) через скалярное произведение (1).

Такой многочлен Φ_0 и называется лучшим приближением данной функции $f = f(x)$ по методу наименьших квадратов (в данном классе функции J).

Если функции $\varphi_0(x), \varphi_1(x), ..., \varphi_n(x)$ линейно независимы на отрезке $[a, b]$, то для данной функции $f(x)$, определенной на $[a, b]$, существует единственный обобщенный многочлен $\Phi_0 = \Phi_0(x)$ наилучшего среднеквадратичного приближения.

Действительно, в силу (3) квадрат расстояния $\rho(f, \Phi)$ от функции $f(x)$ до произвольного обобщенного многочлена $\Phi(x)$ равен

$$\delta = \rho(f, \Phi)^2 = (f - \Phi, f - \Phi) = (f - c_0\varphi_0(x) - c_1\varphi_1(x) - ... - c_n\varphi_n(x), f - c_0\varphi_0(x) - c_1\varphi_1(x) - ... - c_n\varphi_n(x)) = (f, f) + \sum_{i,j=0}^{n} c_i c_j (\varphi_i, \varphi_j) - 2\sum_{i=0}^{n} c_i (f, \varphi_i).$$

Из алгебры известно, что положительно определенная квадратичная функция обладает единственной экстремальной точкой - минимумом функции. Так как в нашем случае расстояние $\delta \geqslant 0$ для всех $c_0, c_1, ..., c_n$, то квадратичная функция δ положительно определена, следовательно минимум δ существует и единствен. Коэффициенты $c_0, ..., c_n$ минимизирующие функцию δ можно определить из системы $\frac{\partial \delta}{\partial c_i} = 0$, $i = 1, 2, ..., n$, или в развернутом виде

$$\begin{cases} (\varphi_0,\varphi_0)c_0 + (\varphi_0,\varphi_1)c_1 + ... + (\varphi_0,\varphi_n)c_n = (f,\varphi_0), \\ (\varphi_1,\varphi_0)c_0 + (\varphi_1,\varphi_1)c_1 + ... + (\varphi_1,\varphi_n)c_n = (f,\varphi_1), \\ . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \\ (\varphi_n,\varphi_0)c_0 + (\varphi_n,\varphi_1)c_1 + ... + (\varphi_n,\varphi_n)c_n = (f,\varphi_n). \end{cases} \tag{5}$$

Здесь, скалярные произведения вычисляются по формуле (1). Определитель этой системы есть определитель Грама и по предположению не равен нулю, поэтому решение системы существует и единственно. Следовательно существует и единствен обобщенный многочлен, минимизирующий расстояние $\rho(f, \Phi)$.

Дискретный случай. Пусть дана табличная функция $f = \{(x_i, f_i)\}$, $i = 0, 1, ..., m$. Рассмотрим класс J обобщенных многочленов порядка n по фиксированной системе функции $\varphi_0(x), \varphi_1(x), ..., \varphi_n(x)$, определенных на отрезке $[x_0, x_m]$. Определим обобщенный многочлен из класса J наилучшего среднеквадратичного приближения для данной табличной функции f. Обозначим через $\overline{\Phi} = \{(x_i, \Phi(x_i))\}$, $\overline{\varphi}_i = \{(x_i, \varphi_i(x_i))\}$, $i = 0, 1, ..., m$, сужение функции $\Phi(x)$ и $\varphi_i(x)$, определенных на отрезке $[x_0, x_m]$, на множество узлов $\{x_0, x_1, ..., x_m\}$.

Обобщенный многочлен $\Phi_0 = \Phi_0(x)$ из класса J является многочленом наилучшего среднеквадратичного приближения для данной табличной функции f, если

$$\rho(f, \overline{\Phi}_0) = \min_{\Phi = \Phi(x) \in J} \rho(f, \overline{\Phi}),$$

где расстояние ρ вычисляется по формуле (3) через скалярное произведение (2).

Такой многочлен $\Phi_0(x)$ и называется лучшим приближением данной табличной функции f по методу наименьших квадратов (в данном классе функции J).

Если функции $\varphi_0(x), \varphi_1(x), ..., \varphi_n(x)$ линейно независимы на отрезке $[x_0, x_m]$, то для данной табличной функции $f = \{(x_i, f_i)\}$ существует единственный обобщенный многочлен $\Phi_0(x)$ на отрезке $[x_0, x_m]$ наилучшего среднеквадратичного приближения. Повторяя предыдущий вывод для функции $\delta = \rho(f, \overline{\Phi})^2$, получим, что коэффициенты $c_0, ..., c_n$ определяются из системы (5), в которой вместо $\varphi_i(x)$ стоят $\overline{\varphi}_i$, а скалярные произведения вычисляются по формуле (2).

В примере ниже рассматриваются обобщенные многочлены по системе функций $\varphi_i(x) = x^i$, $i = 0, 1, ..., n$, то есть обычные многочлены.

Рассмотрим два примера. Выполним команду необходимую для повторного запуска программы:

In[1]:= **Clear[f, x]**

и рассмотрим функцию

In[2]:= **f[x_] = Sin[x]**

Out[1]= sin(x)

Зададим концы отрезка $[a, b]$, число точек n и шаг следования точек h.

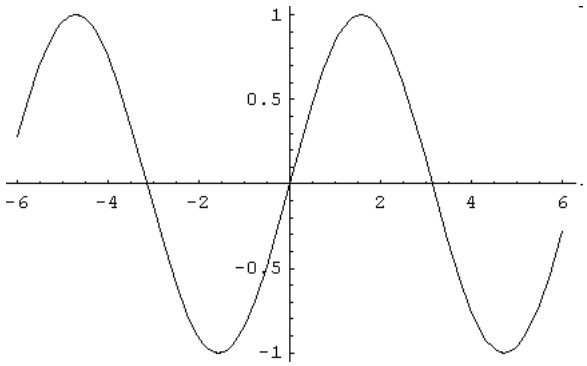
In[3]:= **a = −6; b = 6; n = 7; h = (b − a)/n;**

Сначала построим с помощью функции $f(x)$ табличную функцию и для нее построим многочлен степени, например, 4 наилучший в смысле метода наименьших квадратов (дискретный случай). Затем аппроксимируем непрерывную функцию $f(x)$ на отрезке $[a, b]$ многочленом 5-ой степени по методу наименьших квадратов (непрерывный случай).

Построим график функция

In[4]:= **gr3 = Plot[f[x], {x, a, b}]**

Out[4]=



Дискретный случай. По данной непрерывной функции построим табличную функцию $f1$ и список ее значений $f0$:

In[5]:= **f0 = Table[f[k], {k, a, b, h}]//N**
f1 = Table[{k, f[k]}, {k, a, b, h}]//N

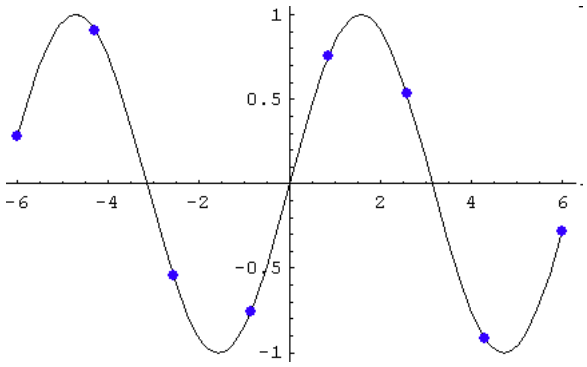
Out[5]= {0.279415, 0.910347, -0.53977, -0.755975, 0.755975, 0.53977, -0.910347, -0.279415}

Out[6]= { {-6, 0.2794,} {-4.2857, 0.9103}, {-2.5714, -0.5397}, {-0.8571, -0.7559}, {0.8571, 0.7559}, {2.5714, 0.5397}, {4.2857,-0.9103}, {6, -0.2794}}

Построим график таблично заданной функции и совместный график.

In[7]:= **gr1 = ListPlot[f1, PlotStyle → {Hue[.7], PointSize[0.02]}];**
gr2 = Show[gr3, gr1]

Out[7]=



Сформулируем задание: для таблично заданной функции **f1** построить многочлен степени **m** наилучшего приближения по методу наименьших квадратов. Построить графики многочлена и общий график всех функции.

Введем степень многочлена

In[8]:= **m = 4;**

Составим список точек, над которыми задана функция **f1** :

In[9]:= **x = Table[k, {k, a, b, h}]//N;**

Введем систему функций φ_i:

In[10]:= **$\varphi_{i\_}$:= x^i** (*Здесь индекс - аргумент функции.*)
φ_0 = Table[1, {k, a, b, h}];

Значение функции φ_0 пришлось выделить отдельной строкой, чтобы избежать возведения нуля в нулевую степень.

Задаем матрицу системы (5); точкой обозначено скалярное произведение векторов по формуле (2).

In[11]:= $\mathbf{p = Table[\varphi_i.\varphi_j, \{i, 0, m\}, \{j, 0, m\}]}$

Out[11]=

$$\begin{pmatrix} 8 & 0. & 123.4285 & 0. & 3355.2419 \\ 0. & 123.4285 & 0. & 3355.2419 & 0. \\ 123.4285 & 0. & 3355.2419 & 0. & 106283.7869 \\ 0. & 3355.2419 & 0. & 106283.7869 & 0. \\ 3355.2419 & 0. & 106283.7869 & 0. & 3.5906 \times 10^6 \end{pmatrix}$$

Зададим столбец свободных членов системы (5)

In[12]:= $\mathbf{bb = Table[f0.\varphi_j, \{j, 0, m\}]}$

Out[12]= {0., -7.08404, 0., -244.72, 0.}

Решаем систему с матрицей **m** и столбцом свободных членов **bb**, используя встроенную функцию.

In[13]:= $\mathbf{s = LinearSolve[p, bb]}$

Out[13]= $\{1.51292 * 10^{-16}, 0.0366378, 0., -0.00345912, 9.73062 * 10^{-19}\}$

Составим многочлен наилучшего приближения:

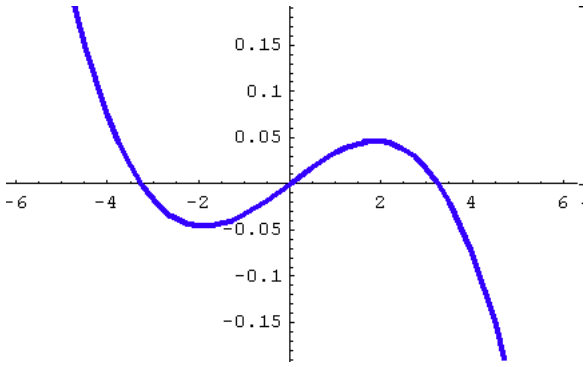
In[14]:= $\mathbf{x = .;\ \ mnk[x\_] = Sum[s[[i + 1]]\ x^i, \{i, 0, m\}]}$

Out[14]= $9.73062 * 10^{-19}x^4 - 0.00345912x^3 - 3.63892 * 10^{-17}x^2 + 0.0366378\ x + 1.51292 * 10^{-16}$

Построим график многочлена наилучшего приближения:

In[15]:= $\mathbf{gr = Plot[mnk[x], \{x, a, b\}, PlotStyle \rightarrow \{Hue[0.7], Thickness[0.01]\}]}$

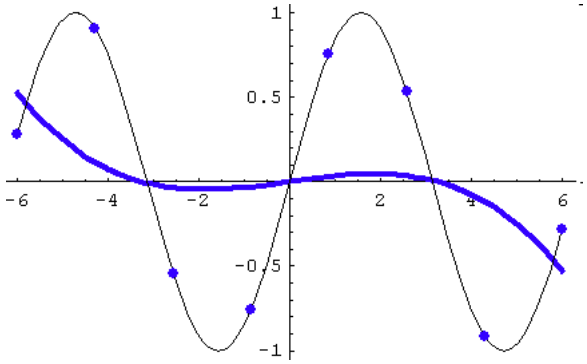
Out[15]=



Построим совмещенный график:

In[16]:= $\mathbf{Show[gr, gr3, gr1]}$

Out[16]=



Таким образом, построенный многочлен четвертого порядка является самым близким в смысле метода наименьших квадратов как к табличной функции $f1$, так и к дифференцируемой функции $Sin(x)$. Причем, в первом случае многочлен является интерполяционным многочленом для дискретной функции $f1$, а во-втором, аппроксимационным многочленом для функции $sin(x)$.

Непрерывный случай. Первые четыре команды выполнены выше.

Введем степень многочлена

In[5]:= $\mathbf{m = 6;}$

и построим аппроксимационный многочлен 6-ой степени для функции $f(x) = sin(x)$ на $[a, b]$ по методу наименьших квадратов.

Введем систему функций $\varphi_{\mathbf{i}}$:

In[6]:= $\varphi_{\mathbf{i\_}} := \mathbf{x^i}$

$\varphi_{\mathbf{0}} = \mathbf{1};$

Задаем матрицу системы (5). Для этого вычисляем скалярные произведения по формуле (1):

In[7]:=

$$\mathbf{m = Table[}\int_{\mathbf{-6}}^{\mathbf{6}} \varphi_{\mathbf{i}}\,\varphi_{\mathbf{j}}\,\mathbf{dx}, \mathbf{\{i, 0, m\}, \{j, 0, m\}//N}$$

Out[7]=

$$\begin{pmatrix} 12. & 0. & 144. & 0. & 3110.4 & 0. \\ 0. & 144. & 0. & 3110.4 & 0. & 79981.7 \\ 144. & 0. & 3110.4 & 0. & 79981.7 & 0. \\ 0. & 3110.4 & 0. & 79981.7 & 0. & 2.23949 \times 10^6 \\ 3110.4 & 0. & 79981.7 & 0. & 2.23949 \times 10^6 & 0. \\ 0. & 79981.7 & 0. & 2.23949 \times 10^6 & 0. & 6.59631 \times 10^7 \end{pmatrix}$$

Вычисляем столбец свободных членов системы (5):

In[8]:=

$$\mathbf{bb = Table[}\int_{\mathbf{-6}}^{\mathbf{6}} \mathbf{f[x]}\,\varphi_{\mathbf{i}}\,\mathbf{dx}, \mathbf{\{i, 0, m\}]//N}$$

Out[8]= {0.,-12.0809,0.,-402.662,0.,-10500.6}

Решаем систему с матрицей **m** и столбцом свободных членов **bb** :

In[9]:= $\mathbf{s = LinearSolve[p, bb]}$

Out[9]= {0.,0.700156,0.,-0.081703,0.,0.00176572}

Составим многочлен наилучшего приближения:

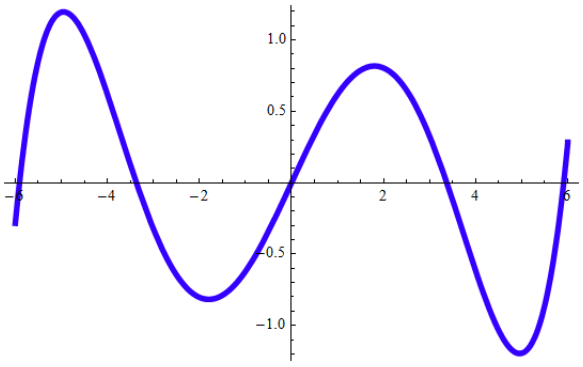
In[10]:= $\mathbf{x = .;\ \ mnk[x\_] = Sum[s[[i + 1]]\ x^i, \{i, 0, m\}]}$

Out[10]= $0.00176572x^5 + 0.x^4 - 0.081703x^3 + 0.x^2 + 0.700156x + 0.$

Построим график многочлена наилучшего приближения:

In[11]:= $\mathbf{gr4 = Plot[mnk[x], \{x, a, b\}, PlotStyle \rightarrow}$
$\mathbf{\{Hue[0.7], Thickness[0.01]\}]}$

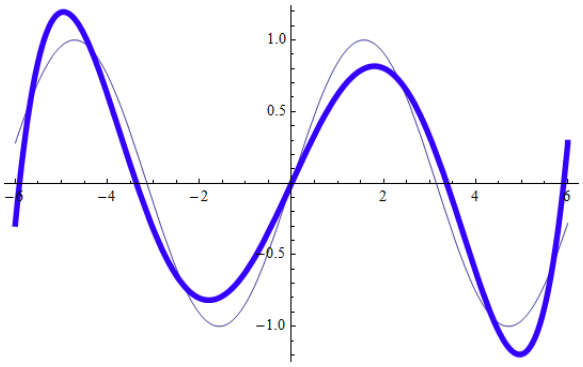
Out[11]=



Построим совмещенный график:

In[12]:= **Show[gr3, gr4]**

Out[12]=



§18 Численное интегрирование

Рассмотрим три метода приближенного вычисления определенного интеграла

$$J = \int_a^b f(x)dx.$$

Разобьем интервал интегрирования точками $a = x_0 < x_1 < ... < x_n = b$ и, используя аддитивностью интеграла, запишем, что

$$J = \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} f(x)dx. \qquad (1)$$

Каждый интеграл

$$r_i = \int_{x_i}^{x_{i+1}} f(x)dx,$$

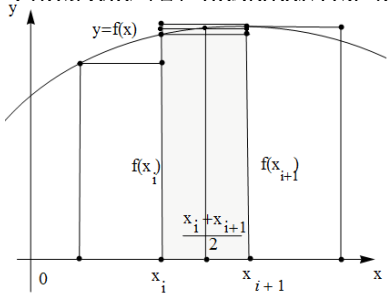
$i = 0, 1, ..., n-1$, в правой части можно считать площадью криволинейной трапеции, ограниченной графиком функции на отрезке $[x_i, x_{i+1}]$, $i = 0, 1, ..., n-1$, (см. геометрический смысл определенного интеграла). В противном случае подобрать константу

$c >$ такую, чтобы $f(x) + c \geqslant 0$ на отрезке $[a, b]$. Определенный интеграл от такой функции будет только константой, равной $(b-a)c$, отличаться от искомого интеграла. Численное вычисление интеграла (1) заключается в замене в (1) площади криволинейной трапеции r_i площадью более простой фигуры, например, площадью прямоугольника с основанием $[x_i, x_{i+1}]$ и высотой $f(x_i)$ ($f(x_{i+1})$ или $f(\frac{x_i+x_{i+1}}{2})$), площадью соответствующей трапеции или криволинейной трапеции, ограниченной параболой. Так полученные формулы приближенного вычисления определенного интеграла называются квадратурными формулами. Рассмотрим эти случаи.

Формула прямоугольников. Пусть шаг следования точек разбиения отрезка равен $h = \frac{b-a}{n}$, то есть $x_i - x_{i-1} = h$ для всех значений i. Заменим каждый интеграл r_i в (1) площадью прямоугольника с основанием h и высотой $f(x_i)$, то есть $hf(x_i)$. Получим следующую формулу приближенного вычисления определенного интеграла

$$J \approx J_h = h \sum_{i=0}^{n-1} f(x_i). \tag{2}$$

Формула (2) называется формулой левых прямоугольников. Аналогично получаем формулы правых прямоугольников и центральных прямоугольников.



$$J_h = h \sum_{i=0}^{n-1} f(x_{i+1}), \qquad J_h = h \sum_{i=0}^{n-1} f(\frac{x_i + x_{i+1}}{2}).$$

Оценим погрешность приближенного вычисления определенного интеграла J по формуле прямоугольников. То есть, оценим разность $R(h) = |J - J_h|$. Для этого рассмотрим формулу (2), остальные формулы рассматриваются аналогично. Дополнительно будем предполагать, что подинтегральная функция $f(x)$ непрерывно дифференцируема. В силу аддитивности интеграла

$$R(h) = |\int_a^b f(x)dx - h \sum_{i=0}^{n-1} f(x_i)| = |\sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} f(x)dx - h \sum_{i=0}^{n-1} f(x_i)|.$$

Отсюда получим, что

$$R(h) \leqslant \sum_{i=0}^{n-1} \Big| \int_{x_i}^{x_{i+1}} f(x)dx - hf(x_i) \Big|. \tag{3}$$

Оценим

$$r_i(h) = \int_{x_i}^{x_{i+1}} f(x)dx - hf(x_i) = \int_{x_i}^{x_i+h} f(x)dx - hf(x_i).$$

Применяя формулу Лагранжа, запишем

$$\frac{dr_i(h)}{dh} = f(x_i + h) - f(x_i) = hf'(\xi),$$

где $x_i \leqslant \xi \leqslant x_{i+1}$. Проинтегрируем это равенство от 0 до h. Учитывая что $r_i(0) = 0$, получим

$$|r_i(h)| = |\int_0^h hf'(\xi)dt| \leqslant \int_0^h h|f'(\xi)|dt \leqslant M\frac{h^2}{2},$$

где $M = \max_{x\in[a,b]} |f'(x)|$. Отсюда и из (3)

$$R(h) \leqslant \sum_{i=0}^{n-1} M\frac{h^2}{2} = M\frac{h^2}{2}n = M\frac{h^2}{2}\frac{b-a}{h} = Ah,$$

где $A = M\frac{b-a}{2}$. Таким образом, получена следующая оценка: $R(h) = |J - J_h| < Ah$.

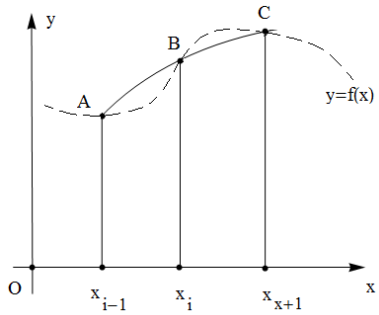
Формула трапеции. Заменим каждый интеграл r_i в (1) площадью трапеции с высотой h и основаниями длины $f(x_i)$ и $f(x_{i+1})$ то есть $\frac{f(x_i)+f(x_{i+1})}{2}h$, получим следующую формулу приближенного вычисления определенного интеграла

$$J \approx J_h = \sum_{i=0}^{n-1} \frac{f(x_i) + f(x_{i+1})}{2}h$$

или

$$J \approx J_h = \frac{h}{2}(f(a) + f(b) + 2\sum_{i=1}^{n-1}(f(x_i)) \tag{4}$$

Формула (4) называется формулой трапеций. Погрешность приближенного вычисления



определенного интеграла по этой формуле, при условии, что подинтегральная функция дважды непрерывно дифференцируема, оценивается величиной $R(h) = |J - J_h| < Ah^2$. Вывод такой оценки подобен аналогичному выводу для формулы прямоугольников.

Формула парабол (формула Симпсона). Интегралы в (1) заменим площадями соответствующих криволинейных трапеций, ограниченных параболами. Разобьем отрезок интегрирования $[a, b]$ на четное число отрезков точками $a = x_0 < x_1 < ... < x_{2n} = b$; шаг следования точек $h = \frac{b-a}{2n}$. Возьмем три последовательных узла x_{i-1}, x_i, x_{i+1}. Пусть $A(x_{i-1}, f(x_{i-1}))$, $B(x_i, f(x_i))$ и $C(x_{i+1}, f(x_{i+1}))$ три точки на графике функции $y = f(x)$ (пунктирная линия).

Покажем сначала, что через точки A, B, C проходит единственная парабола (возможно вырожденная в прямую). Пусть $y = ax^2 + bx + c$ - уравнение параболы. Потребуем, чтобы эта парабола проходила через точки A, B, C, получим следующую систему уравнений относительно коэффициентов a, b и c:

$$\begin{cases} ax_{i-1}^2 + bx_{i-1} + c = f(x_{i-1}), \\ ax_i^2 + bx_i + c = f(x_i), \\ ax_{i+1}^2 + bx_{i+1} + c = f(x_{i+1}). \end{cases}$$

Определитель этой системы есть определитель Вандермонда, поэтому не равен нулю, а система имеет единственное решение. Следовательно, через данные три точки проходит единственная парабола.

Найдем площадь криволинейной трапеции, ограниченной параболой, проходящей через точки A, B, C и стороной $[x_{i-1}, x_{i+1}]$:

$$s_i = \int_{x_{i-1}}^{x_{i+1}} (ax^2 + bx + c)dx = \int_{x_i-h}^{x_i+h} (ax^2 + bx + c)dx = (a\frac{x^3}{3} + b\frac{x^2}{2} + cx)\Big|_{x_i-h}^{x_i+h} =$$

$$a\frac{(x_i+h)^3}{3}+b\frac{(x_i+h)^2}{2}+c(x_i+h)-(a\frac{(x_i-h)^3}{3}+b\frac{(x_i-h)^2}{2}+c(x_i-h))=$$

$$2x_i^2ha+\frac{2}{3}ah^3+2bx_ih+2ch=2h\Big[\frac{a}{3}(3x_i^2+h^2)+bx_i+c\Big]. \quad (5)$$

Найдем выражение (5) другим способом. Так как $f(x_{i-1})=f(x_i-h)=a(x_i-h)^2+b(x_i-h)+c$, $f(x_i)=ax_i^2+bx_i+c$, $f(x_{i+1})=f(x_i+h)=a(x_i+h)^2+b(x_i+h)+c$, то

$$\frac{h}{3}[f(x_{i-1})+4f(x_i)+f(x_{i+1})]=\frac{h}{3}[a(x_i-h)^2+b(x_i-h)+c+$$

$$4(a(x_i)^2+b(x_i)+c)+a(x_i+h)^2+b(x_i+h)+c=2h\Big[\frac{a}{3}(3x_i^2+h^2)+bx_i+c\Big].$$

Сравнивая с (5), получим равенство

$$s_i=\frac{h}{3}[f(x_{i-1})+4f(x_i)+f(x_{i+1})].$$

Разобьем теперь отрезок $[a,b]$ на отрезки $[x_{i-1},x_{i+1}]$, $i=1,3,5,...,n-1$, над каждым отрезком построим криволинейную трапецию ограниченную параболой, как это было сделано выше, и просуммируем площади таких криволинейных парабол, получим, что

$$J\approx J_h=\frac{h}{3}[f(x_0)+4f(x_1)+f(x_2)+f(x_2)+4f(x_3)+f(x_4)+...+$$

$$f(x_{2n-4})+4f(x_{2n-3})+f(x_{2n-2})+f(x_{2n-2})+4f(x_{2n-1})+f(x_{2n})].$$

Или

$$J\approx J_h=\frac{h}{3}(f(a)+f(b)+4\sum_{i=0}^{n-1}f(x_{2i+1})+2\sum_{i=1}^{n-1}f(x_{2i})). \quad (6)$$

Формула (6) называется формулой парабол. Погрешность приближенного вычисления определенного интеграла по этой формуле, при условии, что подинтегральная функция четырежды непрерывно дифференцируема, оценивается величиной $R(h)=|J-J_h|<Ah^4$.

Отметим частный случай формулы (6) - интервал интегрирования разбит на 2 интервала с шагом $h=\frac{b-a}{2}$:

$$J\approx J_h=\frac{h}{3}(f(x_0)+4f(x_1)+f(x_2).$$

Практическая оценка погрешности. Допуская некоторую вольность, будем считать, что полученные оценки погрешности квадратичных формул точные:

$$R(h)=|J-J_h|=A\,h^k,$$

где $k=1,2,4$ для соответственно для формулы прямоугольников, трапеций и формулы Симпсона. Отсюда $J=J_h\pm A\,h^k$, а, удваивая шаг, получим $J=J_{2h}\pm A\,(2h)^k$. Приравняем правые части и, после преобразований, получим

$$R(h)=Ah^k\leqslant\frac{|J_h-J_{2h}|}{2^k-1}.$$

Такая оценка позволяет в практических вычислениях пользоваться следующим правилом: приближенное значение определенного интеграла J_h будет иметь предельную абсолютную погрешность $R(h)$ меньше ε, если

$$\frac{|J_h - J_{2h}|}{2^k - 1} < \varepsilon, \tag{7}$$

где $k = 1, 2, 4$ соответственно для формулы прямоугольников, трапеций и формулы Симпсона.

Рассмотрим пример численного интегрирования. Зададим функцию

In[1]:= $\mathbf{f[x\_] = Sin[x^2 + 0.2]}$;

Введем концы отрезка $[a, b]$:

In[2]:= $\mathbf{a = 1}$;

$\mathbf{b = \frac{\pi}{2}}$;

Требуется наити приближенное значение определенного интеграла от функции $f[x]$ тремя рассмотренными методами. Погрешность результата не должна превышать

In[3]:= $\varepsilon = \mathbf{0.001}$

Для сравнения воспользуемся встроенной в программу Mathematica функцию вычисления определенного интеграла

In[4]:= $\mathbf{\int_1^{\pi/2} f[x]dx}$

Out[4]= 0.496522

n.1 Формула прямоугольников

Запишем правую часть (1) как функцию от числа отрезков n. Так как $x_i = a + i\ h = a + \frac{b-a}{n} i$, то

In[5]:= $\mathbf{J[n\_] := \frac{b-a}{n} \sum_{i=0}^{n-1} f[a + \frac{b-a}{n} i]}$

Рассмотрим следующий цикл, в котором проверяется точность вычисления определенного интеграла по формуле (7) в зависимости от числа отрезков n.

In[6]:=
```
Do[If[Abs[J[n] - J[2n]] < ε,
    Print["Интеграл  J = ", J[n], "при  n = ", n]; Break[]],
  {n, 1, 100}]
```

Out[6]= Интеграл J=0.498526 при n=67

n.2 Формула трапеции

Запишем правую часть (2) как функцию от числа отрезков n:

In[7]:= $\mathbf{J[n\_] := \frac{b-a}{2n}(f[a] + f[b] + 2 \sum_{i=1}^{n-1} f[a + \frac{b-a}{n} i])}$

и рассмотрим следующий цикл:

In[8]:=
```
Do[If[Abs[J[n] - J[2n]]/3 < ε,
    Print["Интеграл  J = ", J[n], "при  n = ", n]; Break[]],
  {n, 1, 100}]
```

Out[8]= Интеграл J=0.495342 при n=9

n.3 Формула парабол

Запишем правую часть (3) как функцию от числа отрезков 2n:

In[9]:= $\mathbf{J[n\_] := \frac{b-a}{6n}(f[a] + f[b] + 4\sum_{i=0}^{n-1} f[a + \frac{b-a}{2n}(2i+1)] + 2\sum_{i=1}^{n-1} f[a + \frac{b-a}{2n}2i])}$

и рассмотрим следующий цикл:

```
In[10]:= Do[If[Abs[J[n] - J[2n]]/15 < ε,
           Print["Интеграл  J = ", J[n], "при  n = ", n]; Break[]],
         {n, 1, 100}]
Out[10]= Интеграл J=0.496598 при n=2
```

n.4 Формулы Ньютона-Котеса

Для численного вычисления определенного интеграла

$$J = \int_a^b f(x)dx \tag{1}$$

можно ввести табуляцию функции $f(x)$ на отрезке $[a, b]$, построить по табличной функции многочлен Лагранжа или Ньютона, на который и заменить подинтегральную функцию. В результате получится семейство квадратурных формул Ньютона-Котеса.

Рассмотрим разбиение отрезка $[a, b]$ точками $a = x_0 < x_1 < ... < x_n = b$; шаг следования точек $h = (b - a)/n$, и для табличной функции $(x_i, f(x_i))$, $i = 0, 1, ..., n$, построим многочлен Лагранжа и подставим в (5) вместо $f(x)$:

$$J \approx \int_a^b \Big(f(x_0)\frac{(x-x_1)(x-x_2)...(x-x_n)}{(x_0-x_1)(x_0-x_2)...(x_0-x_n)} + f(x_1)\frac{(x-x_0)(x-x_2)...(x-x_n)}{(x_1-x_0)(x_1-x_2)...(x_1-x_n)} + ... + f(x_n)\frac{(x-x_0)(x-x_1)...(x-x_{n-1})}{(x_n-x_1)(x_n-x_2)...(x_n-x_{n-1})}\Big)dx.$$

Сделаем замену переменной в этом интеграле по формуле $x = x_0 + qh$. Подинтегральная функция примет вид

$$L_h(x_0+hq) = f(x_0)\frac{(x_0+hq-x_1)...(x_0+hq-x_n)}{(x_0-x_1)...(x_0-x_n)} + ...+ f(x_i)\frac{(x_0+hq-x_0)...(x_0+hq-x_n)}{(x_i-x_0)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)} + ...+ f(x_n)\frac{(x_0+hq-x_0)...(x_0+hq-x_{n-1})}{(x_n-x_0)...(x_n-x_{n-1})}.$$

Так как

$$x_0 + hq - x_i = x_0 - x_i + hq = -hi + hq = h(q-i)$$

и

$$(x_i - x_0)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n) =$$

$$ih(i-1)h...h(-h)...(-(n-i)h) = i!h^i(-1)^{n-i}h^{n-i} = h^n i!(n-i)!,$$

то i - е слагаемое преобразуется следующим образом

$$f(x_i)\frac{(x_0+hq-x_0)...(x_0+hq-x_n)}{(x_i-x_0)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)} =$$

$$f(x_i)\frac{hqh(q-1)...h(q-(i-1))h(q-(i+1))...h(q-n)}{h^n i!(n-i)!} =$$

$$f(x_i)\frac{(-1)^n}{i!(n-i)!}\frac{q(q-1)...(q-n)}{q-i}.$$

Поэтому интеграл

$$J \approx h\int_0^n \sum_{i=0}^{n}\frac{(-1)^{n-i}f(x_i)}{i!(n-i)!}\cdot\frac{q(q-1)\cdot...\cdot(q-n)}{q-i}dq$$

или

$$J \approx (b-a)\sum_{i=0}^{n} H_i f(x_i), \tag{2}$$

где коэффициенты

$$H_i = \frac{1}{n}\cdot\frac{(-1)^{n-i}}{i!(n-i)!}\int_0^n \frac{q(q-1)\cdot...\cdot(q-n)}{q-i}dq \tag{3}$$

зависят только от шага следования точек и их числа, но не от значений функции. Семейство формул (2), зависящих от n, и есть квадратурные формулы Ньютона-Котеса.

Найдем определенный интеграл от функции

In[1]:= **f[x_] = Sin[x^2 + .2];**

на отрезке $[a,b]$,

In[2]:= **a = 1; b = Pi/2;**

Разобьем отрезок $[a,b]$ точками на n отрезков длины h:

In[3]:= **n = 5; h = (b - a)/n;**

и введем координаты точек разбиения:

In[4]:= **Do[x_i = a + h i, {i, 0, n}]**

Для табличной функции

In[5]:= **fs = Table[{x_i, f[x_i]}, {i, 0, n}]//N**

Out[5]=

{{ 1, 0.93203,} { 1.11416, 0.99163,} {1.22831, 0.99049},
{1.34247, 0.90836}, {1.45663, 0.73101}, {1.57079, 0.45661}}

определим универсальные коэффициенты по формуле (3)

In[6]:=

$$\mathbf{H[i\_]} := \frac{1}{n}\cdot\frac{(-1)^{n-i}}{i!(n-i)!}\int_0^n \frac{\prod_{k=0}^{n}(q-k)}{q-i}dq//N$$

Приведем несколько таких коэффициентов:

In[7]:= **Table[H[i], {i, 0, n}]**

Out[7]= {0.0659722,0.260417,0.173611,0.173611,0.260417,0.0659722}

Теперь по формуле (2) получаем приближенное значение определенного интеграла

In[8]:= **(b − a) $\sum_{i=0}^{n}$ H[i]f[x_i]**

Out[8]= 0.496525

Заметим, что для для вычисления определенного интеграла для другой функции на том же интервале и с тем же числом точек разбиения, достаточно найти значения новой функции в узлах и пересчитать предыдущую сумму.

n.5 Квадратурные формулы Гаусса-Кристоффеля

Рассмотрим приближенное вычисление интеграла от произведения данной функции $f(x)$ и некоторой весовой функции $p(x)$ в следующем виде:

$$\int_a^b p(x)f(x)dx \approx \sum_{i=1}^{n} A_i f(x_i). \tag{1}$$

Весовые коэффициенты A_i и узлы x_i, принадлежащие интервалу (a, b), $i = 0, 1, ..., n$, подбираются так, чтобы формула (1) имела алгебраический порядок точности, то есть была бы точна для многочленов определенного порядка, взятых в качестве функции $f(x)$. Весовая функция должна быть интегрируема на данном интервале (конечном или бесконечном). При определении весовых коэффициентов используются корни ортогональных многочленов.

Семейство многочленов $\{d_0(x), d_1(x), ..., d_n(x), ...\}$ образует ортогональную систему на отрезке $[a, b]$ с весом $p(x)$, если

$$\int_a^b p(x)d_i(x)d_j(x)dx = \begin{cases} 0, & \text{если } i \neq j, \\ \neq 0, & \text{если } i = j. \end{cases} \tag{2}$$

Как известно, (§10.4, [3]) многочлен $d_n(x)$ имеет ровно n действительных корней на промежутке (a, b).

Теорема. Формула (1) точна для произвольного многочлена степени $2n-1$ (имеет алгебраический порядок точности $2n-1$), если ее узлами x_i служат корни многочлена $d_n(x)$, из семейства ортогональных многочленов, на интервале (a, b) с весом $p(x)$ и весовыми коэффициентами

$$A_i = \int_a^b \frac{(x - x_1)...(x - x_{i-1})(x - x_{i+1})...(x - x_n)}{(x_i - x_1)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n)} p(x)dx. \tag{3}$$

При этом формулы (1) называются квадратурными формулами Гаусса-Кристоффеля.

Приведем три примера семейств ортогональных многочленов и покажем их применение при вычислении определенных интегралов.

Многочлены Лагерра $L_n(x)$, $n = 0, 1, ...$, определяются как многочлены, ортогональные на промежутке $[0, \infty)$ с весовой функцией $p(x) = e^{-x}$:

$$\int_0^\infty e^{-x} L_i(x) L_j(x) dx = \begin{cases} 0, & \text{если } i \neq j, \\ (i!)^2, & \text{если } i = j. \end{cases}$$

Многочлены Лагерра удовлетворяют следующему рекуррентному соотношению

$$L(0,x) = 1, \qquad L(0,x) = -x + 1,$$

$$L(n,x) - (2n - 1 - x)L(n-1,x) + (n-1)^2 L(n-2,x) = 0,$$

которое программно реализуется командами

In[1]:= $\mathbf{L[0,x] = 1;}$
$\mathbf{L[1,x] = -x + 1;}$
$\mathbf{L[n\_,x\_] := L[n,x] = (2n - 1 - x)L[n-1,x] - (n-1)^2 L[n-2,x]}$

Теперь многочлен Лагерра, например, при

In[2]:= $\mathbf{n = 9;}$

можно получит командой

In[3]:= $\mathbf{L[n,x]//Expand}$

Out[3]= $-x^9 + 81\ x^8 - 2592\ x^7 + 42336\ x^6 - 381024\ x^5 + 1905120\ x^4 - 5080320\ x^3 + 6531840\ x^2 - 3265920\ x + 362880$

Найдем корни многочлена $L[n,x]$:

In[4]:= $\mathbf{s = x/.Solve[L[n,x] == 0,x]}$

Out[4]= {0.152322, 0.80722, 2.00514, 3.78347, 6.20496, 9.37299, 13.4662, 18.8336, 26.3741}

и введем узлы

In[5]:= $\mathbf{Table[x_i = s[[i]],\{i,1,n\}];}$

Коэффициенты (3) вычисляются по формулам:

In[6]:= $\mathbf{g[x\_] = L[n-1,x]//Expand;}$

$$\mathbf{A[i\_] = \left(\frac{(n-1)!}{n\ g[x_i]}\right)^2 x_i;}$$

Составим список весовых коэффициентов:

In[6]:= $\mathbf{Table[A[i],\{i,1,n\}]}$

Out[6]= $\{0.33615, 0.41121, 0.19929, 0.047461, 0.0056, 0.00031, 6.59212 \times 10^{-6}, 4.11077 \times 10^{-8}, 3.29087 \times 10^{-11}\}$

Теперь можно воспользоваться формулой (1) и найти приближенное значение, например, такого интеграла:

$$\int_0^\infty f(x)dx = \int_0^\infty e^x e^{-x} f(x)dx \approx \sum_{i=1}^n A_i e^{x_i} f(x_i).$$

Вычислим последнюю сумму для функции $f(x) = 1/(1+x^2)$:

In[7]:= $\mathbf{\sum_{i=1}^{n} A[i]\ E^{x_i}\ \frac{1}{1+x_i^2}//Chop}$

Out[7]= 1.53878

Точное значение этого интеграла равно $\pi/2 \approx 1.5708$.

Многочлены Эрмита $H_n(x)$, $n = 0, 1, ...$, определяются как многочлены, ортогональные на всей числовой оси с весовой функцией $p(x) = e^{-x^2}$:

$$\int_{-\infty}^{\infty} e^{-x^2} H_i(x) H_j(x) dx = \begin{cases} 0, & \text{если } i \neq j, \\ 2^i i! \sqrt{\pi}, & \text{если } i = j. \end{cases}$$

Многочлены Эрмита удовлетворяют рекуррентному соотношению вида

$$H(0,x) = 1, \qquad H(0,x) = 2x,$$

$$H(n,x) - 2xH(n-1,x) + 2(n-1)H(n-2,x) = 0,$$

которое программно реализуется такими командами:

In[1]:=
```
H[0, x] = 1;
H[1, x] = 2x;
H[n_, x_] := H[n, x] = 2x H[n - 1, x] - 2(n - 1) H[n - 2, x]
```

Найдем корни многочлена $H[9,x]$:

In[2]:=
```
n = 9;
se = Sort[x /. Solve[H[n, x] == 0, x]//N//Chop]
```

Out[2]=
```
{-3.19099, -2.26658, -1.46855, -0.723551,
0, 0.723551, 1.46855, 2.26658, 3.19099}
```

и введем узлы

In[5]:=
```
Table[x_i = se[[i]], {i, 1, n}];
```

Коэффициенты (3) вычисляются по формулам:

In[6]:=
```
g[x_] = H[n - 1, x]//Expand;
```

$$\mathbf{A[i\_]} = \frac{\mathbf{2^{n-1}(n-1)!\sqrt{\pi}}}{\mathbf{n\ g[x_i]^2}}$$

Составим список весовых коэффициентов:

In[6]:=
```
Table[A[i], {i, 1, n}]
```

Out[6]=
```
{0.720235, 0.432652, 0., 0.047461, 0.0056, 0.00031,
 6.59212 × 10^-6, 4.11077 × 10^-8, 3.29087 × 10^-11}
```

Теперь можно воспользоваться формулой (1) и найти приближенное значение интеграла:

$$\int_{-\infty}^{\infty} f(x)dx = \int_{-\infty}^{\infty} e^{-x^2} e^{x^2} f(x)dx \approx \sum_{i=1}^{n} A_i e^{x_i^2} f(x_i).$$

Вычислим последнюю сумму для функции $f(x) = 1/(1+x^2)$:

In[7]:=
```
Sum[A[i] E^(x_i^2) 1/(1 + x^2), {i, 1, n}]//Chop
```

Out[7]=
```
2.62494
```

Точное значение несобственного интеграла от этой функции равно $\pi \approx 3.14159$.

Отметим, что, в соответствии с теоремой, формула (1) точна для многочленов степени не большей $2n-1$. Этот факт можно просто проиллюстрировать. Сгенерируем многочлен степени, например, $2n-1$:

In[8]:=
```
f[x_] = Sum[Random[Integer, {-10, 10}] x^i, {i, 1, 2n - 1}]
```

Out[8]=
```
-10 x^17 + x^16 - 6 x^15 - 8 x^14 - 6 x^13 + 2 x^12 + 4 x^11 - 2 x^10
```

$$-6\ x^9 - 10\ x^8 + 5\ x^7 + 9\ x^6 + 4\ x^5 + 10\ x^4 + 8\ x^3 + 9\ x^2 + x + 3$$

и вычислим части равенства (1). Левую часть найдем, используя встроенную функцию:

```
In[9]:= ∫_{-∞}^{∞} e^{-x^2} f[x] dx
Out[9]= -524.362
```

Правая часть (1):

```
In[10]:= Σ_{i=1}^{n} A[x_i] f[x_i]
Out[10]= -524.362
```

Квадратурные формулы Эрмита - это частный случай формул (1) при $a = -1$, $b = 1$ и весовой функции $p(x) = 1/\sqrt{1-x^2}$.

Системой ортогональных многочленов на промежутке $[-1, 1]$ будет система, состоящая из многочленов Чебышева $T_n(x)$ (см. §17, n.2) с весовой функцией $p(x) = 1/\sqrt{1-x^2}$.

Возьмем в качестве узлов на промежутке $[-1, 1]$ корни многочлена Чебышева $T_n(x)$:

$$x_i = \cos\frac{2i-1}{2n}\pi. \tag{4}$$

Коэффициенты (3) в этом случае явно вычисляются и равны

$$A_i = \frac{\pi}{n}, \quad i = 1, ..., n.$$

Теперь формула (1) примет вид

$$\int_{-1}^{1} \frac{f(x)}{\sqrt{1-x^2}}dx \approx \frac{\pi}{n}\sum_{i=1}^{n} f(x_i) \tag{5}$$

и известна она как *квадратурная формула Мелера.*

Для вычисления интеграла от функции $f(x)$ на интервале (a, b) надо сначала сделать замену переменной под знаком интеграла по формуле

$$x = \frac{b+a}{2} + \frac{b-a}{2}t$$

и воспользоваться формулой Мелера следующим образом:

$$\int_a^b f(x)dx = \frac{b-a}{2}\int_{-1}^{1} f(\frac{b+a}{2} + \frac{b-a}{2}t)dt =$$

$$\frac{b-a}{2}\int_{-1}^{1} \frac{f(\frac{b+a}{2} + \frac{b-a}{2}t)\sqrt{1-t^2}}{\sqrt{1-t^2}}dt \approx \tag{6}$$

$$\frac{b-a}{2}\cdot\frac{\pi}{n}\sum_{i=1}^{n}\sqrt{1-t_i^2}f(\frac{b+a}{2} + \frac{b-a}{2}t_i),$$

где t_i - корни многочлена Чебышева $T_n(x)$, найденные по формуле (4).

Приведем пример. Пусть

In[1]:= $\mathbf{n = 30; a = -7; b = 7;}$

Рассмотрим функцию:

In[2]:= $\mathbf{f[x\_] = \frac{1}{1+x^2};}$

Найдем узлы многочлена Чебышева по формуле (4):

In[3]:= $\mathbf{Table[t_i = Cos[(2i-1)\pi/(2n)], \{i, 1, n\}]//N;}$

и вычислим интеграл по формуле (6):

In[4]:= $\mathbf{\frac{b-a}{2} \cdot \frac{\pi}{n} \sum_{i=1}^{n} \sqrt{1-t_i^2} f[\frac{b+a}{2} + \frac{b-a}{2} t_i]//N}$

Out[4]= 2.8567

Для сравнения вычислим интеграл от функции $f[x]$ на отрезке $[a, b]$, используя встроенную функцию:

In[5]:= $\mathbf{\int_a^b f[x]dx//N}$

Out[5]= 2.8578

§19 Численное дифференцирование

Определим производные для табличной функции $y = \{(x_i, y_i)\}$, $i = 0, 1, ..., n$, у которой аргументы (узлы) x_i расположены равномерно с шагом h.

Рассмотрим вспомогательную гладкую функцию $f(x)$ на $[x_0, x_n]$. Применяя формулу Тейлора для такой функции, запишем, что

$$f(x_{i+1}) = f(x_i + h) = f(x_i) + f'(x_i)h + \frac{1}{2}f''(\xi_1)h^2,$$

$$f(x_{i-1}) = f(x_i - h) = f(x_i) - f'(x_i)h + \frac{1}{2}f''(\xi_2)h^2.$$

Отсюда получим

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} - \frac{1}{2}f''(\xi_1)h,$$

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h} - \frac{1}{2}f''(\xi_2)h.$$

Видно, что первые слагаемые справа отличаются от производной гладкой функции в узле x_i на слагаемое порядка h. По аналогии с гладким случаем, определим первые производные табличной функции в узле x_i равенствами

$$y'_i = \frac{y_{i+1} - y_i}{h}, \quad i = 0, 1, ..., n-1,$$

$$y'_i = \frac{y_i - y_{i-1}}{h}, \quad i = 1, 2, ..., n$$

причем первую производную назовем *правой разностной производной*, вторую - *левой разностной производной*. Отметим, что правая разностная производная не определена в последнем узле, левая разностная производная - в первом.

Из разложений

$$f(x_{i+1}) = f(x_i + h) = f(x_i) + f'(x_i)h + \frac{1}{2}f''(x_i)h^2 + \frac{1}{3!}f^{(3)}(\xi_1)h^3,$$

$$f(x_{i-1}) = f(x_i - h) = f(x_i) - f'(x_i)h + \frac{1}{2}f''(x_i)h^2 - \frac{1}{3!}f^{(3)}(\xi_2)h^3$$

получим

$$f(x_{i+1}) - f(x_{i-1}) = 2f'(x_i)h + \frac{1}{3!}(f^{(3)}(\xi_1) + f^{(3)}(\xi_2))h^3.$$

Отсюда

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - \frac{1}{3!}(f^{(3)}(\xi_1) + f^{(3)}(\xi_2))h^2$$

Первое слагаемое справа отличаются от производной гладкой функции в узле x_i на слагаемое порядка h^2. Введем еще одну первую производную табличной функции в узле x_i

$$y'_i = \frac{y_{i+1} - y_{i-1}}{2h}, \quad i = 1, ..., n-1,$$

и назовем *центральной разностной производной*. Центральная разностная производная не определена в граничных узлах.

Из разложений гладкой функции

$$f(x_{i+1}) = f(x_i + h) = f(x_i) + f'(x_i)h + \frac{1}{2}f''(x_i)h^2 + \frac{1}{3!}f^{(3)}(x_i)h^3 + \frac{1}{4!}f^{(4)}(\xi_1)h^4,$$

$$f(x_{i-1}) = f(x_i - h) = f(x_i) - f'(x_i)h + \frac{1}{2}f''(x_i)h^2 - \frac{1}{3!}f^{(3)}(x_i)h^3 + \frac{1}{4!}f^{(4)}(\xi_2)h^4,$$

найдем

$$f(x_{i+1}) + f(x_{i-1}) = 2f(x_i) + f''(x_i)h^2 + \frac{1}{4!}(f^{(4)}(\xi_1) + f^{(4)}(\xi_2))h^4,$$

Отсюда найдем

$$f''(x_i) = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2} - \frac{1}{4!}(f^{(4)}(\xi_1) + f^{(4)}(\xi_2))h^2.$$

Первое слагаемое справа отличаются от второй производной гладкой функции в узле x_i на слагаемое порядка h^2. Определим по аналогии вторую производную табличной функции в узле x_i равенством

$$y''_i = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}, \quad i = 1, ..., n-1,$$

и назовем *второй разностной производной* табличной функции в узле x_i.

Аналогично можно определить разностные производные табличной функции высших порядков.

Рассмотрим еще один способ определения производных табличной функции в узле x_i. Рассмотрим многочлен Лагранжа $L_k(x)$ построенный либо по табличной функции y, либо по части табличной функции y, определенной на близких к x_i узлах. Теперь можно положить

$$y_i^{(m)} = L_k^{(m)}(x_i), \quad 0 \leqslant m \leqslant k.$$

Пример. 1). Дана табличная функция

x	0.1	0.2	0.3	0.4	0.5	0.6
y	0.55	0.78	0.73	0.46	0.24	0.16

Найти левую разностную производную в точке 0.4 и вторую разностную производную в точке 0.5.

Так как шаг следования узлов равен $h = 0.1$, то левая разностная производная в точке 0.4 равна $\frac{0.46-0.73}{0.1} = -2.6$, а вторая разностная производная в точке 0.5. $\frac{0.16-2\cdot 0.24+0.46}{0.1^2} = 14$.

§20 Численное решение дифференциальных уравнений

Решаем задачу Коши для дифференциального уравнения первого порядка

$$y'(x) = f(x, y(x)), \quad y(x_0) = y_0 \tag{1}$$

на отрезке $[x_0 = a, b]$.

Разобьем интервал точками $a = x_0 < x_1 < ... < x_n = b$ расположенными с шагом $h = (b-a)/n$. Численным решением задачи Коши (1) является табличная функция $y = \{(x_i, y_i)\}$, для которой известна оценка погрешности $|y_i - y(x_i)|$, где $y(x)$ аналитическое решение задачи (1). Оценка погрешности позволяет судить насколько найденное численное решение удовлетворяет требованиям практической задачи.

Метод Эйлера. Применяя ряд Тейлора функции $y(x)$ в точке x_i, можно записать

$$y(x_{i+1}) = y(x_i + h) = y(x_i) + y'(x_i)h + ch^2$$

где ch^2 - остаточный член в форме Лагранжа. Перепишем это равенство с учетом (1):

$$y(x_{i+1}) = y(x_i) + h\, f(x_i, y(x_i)) + ch^2. \tag{2}$$

При $i = 0$ из (2) и начального условия (1) получим

$$y(x_1) = y(x_0) + h\, f(x_0, y(x_0)) + ch^2 = y_0 + h\, f(x_0, y_0) + ch^2. \tag{3}$$

Обозначим через

$$y_1 = y_0 + h\, f(x_0, y_0) \tag{4}$$

и объявим y_1 вторым значением табличной функции - численным решением задачи Коши, первое значение - y_0. Из (3) и (4) получаем погрешность приближенного значения y_1 :

$$y(x_1) - y_1 = ch^2. \tag{5}$$

При $i = 1$ из (2) и (5) получим

$$y(x_2) = y(x_1) + h\, f(x_1, y(x_1)) + c_1 h^2 = y_1 + ch^2 + h\, f(x_1, y_1 + ch^2) + c_1(h^2)$$

Применяя формулу Тейлора, запишем

$$y(x_2) = y_1 + ch^2 + h\,[f(x_1, y_1) + f'_y(x_1, y_1)ch^2 + \widetilde{c}h^4] + c_1 h^2$$

или, пренебрегая достаточно малыми слагаемыми с h^3 и h^5, получим

$$y(x_2) = y_1 + h\,f(x_1, y_1) + ch^2 + c_1h^2 = y_1 + h\,f(x_1, y_1) + 2c_2h^2, \tag{6}$$

где $c_2 = (c + c_1)/2$. Обозначим через

$$y_2 = y_1 + h\,f(x_1, y_1) \tag{7}$$

и объявим y_2 третьим значением табличной функции - численным решением задачи Коши. Из (6) и (7) получаем погрешность приближенного значения y_2 :

$$y(x_2) - y_2 = 2c_2h^2. \tag{8}$$

И так далее, другими словами, все численные решения задачи Коши будем вычислять по формуле

$$y_{i+1} = y_i + h\,f(x_i, y_i), \tag{9}$$

при $i = 0, 1, ..., n-1$. При этом, глобальная погрешность найденного численного решения

$$y(x_n) - y_n = nc_nh^2 = \frac{b-a}{h}c_nh^2 = (b-a)c_nh.$$

Таким образом, метод Эйлера имеет глобальную погрешность порядка h и поэтому практически не применяется. Можно увеличить число членов ряда Тейлора в рассуждениях выше, тем самым повысить точность метода. Но, в этом случае возникнет необходимость вычислять производные функции $f(x, y)$. Следующий сильный метод позволяет избежать таких трудностей и имеет достаточно высокую точность.

Отметим, что данный метод является k-шаговым, если при определении приближенного значения y_{i+1} требуются k предыдущих значений y_i, y_{i-1}, ..., y_{i-k+1}. В этом смысле метод Эйлера является одношаговым методом, из формулы (9) видно, что для определения приближенного значения y_{i+1} требуется только одно предыдущее значение.

Метод Рунге-Кутты. Будем искать решение задачи Коши с помощью следующей рекуррентной формулы

$$y_{i+1} = y_i + h\,\varphi(x_i, y_i, h), \tag{10}$$

где правая часть в (10) является приближением отрезка ряда Тейлора функции $y(x)$ в точке x_i до p-го порядка *и не содержит производных*. Рассмотрим наиболее простые случаи.

При $p = 1$, формула (9) представляет метод Рунге-Кутты первого порядка - правая часть (9) не содержит производных и приближает отрезок ряда Тейлора функции $y(x)$ в точке x_i до первой производной (см. формулы (4) и (2), например).

Видим, что в этом случаем метод Рунге-Кутты первого порядка совпадает с методом Эйлера.

Пусть $p = 2$. Будем искать функцию в (10) в следующем виде

$$\varphi(x, y, h) = c_1f(x, y) + c_2f(x + \alpha h, y + \beta hf(x, y))).$$

Следовательно формулу (10) в виде

$$y_{i+1} = y_i + h\,[c_1f(x_i, y_i) + c_2f(x_i + \alpha h, y_i + \beta hf(x_i, y_i)))]. \tag{11}$$

Для определения параметров c_1, c_2, α, β преобразуем равенство (11). Применяя формулу Тейлора, запишем

$$f(x+\alpha h, y+\beta h f(x,y)) = f(x,y) + f'_x(x,y)\alpha h + f'_y(x,y)\beta h f(x,y) + o(h^2)$$

Тогда равенство (11) примет следующий вид

$$y_{i+1} = y_i + h\left[(c_1+c_2)f(x_i,y_i) + h(c_2\alpha f'_x(x_i,y_i) + c_2\beta f'_y(x_i,y_i)f(x_i,y_i))\right] + o(h^3). \quad (12)$$

Применяя теперь ряд Тейлора для функции $y(x)$ в точке x_i до вторых производных, запишем

$$y(x_{i+1}) = y(x_i+h) = y(x_i) + y'(x_i)h + \frac{h^2}{2}y''(x_i)h^2 + o(h^3). \quad (13)$$

По (1) $y'(x) = f(x,y(x))$, тогда $y''(x) = f'_x(x,y(x)) + f'_y(x,y(x))y'(x) = f'_x(x,y(x)) + f'_y(x,y(x))f(x,y(x))$. Поэтому (13) примет вид

$$y(x_{i+1}) = y(x_i) + h[f(x_i,y(x_i)) + \frac{h}{2}f'_x(x_i,y(x_i)) + \frac{h}{2}f'_y(x_i,y(x_i))f(x_i,y(x_i))] + o(h^3). \quad (14)$$

Сравнивая правые части равенств (12) и (14), видим, что они совпадают с точностью до бесконечно малых порядка h^3 и замены $y(x_i) \to y_i$, если

$$c_1 + c_2 = 1, \qquad 2c_2\alpha = 1, \qquad 2c_2\beta = 1$$

Пусть $c_2 = \lambda$, тогда $c_1 = 1-\lambda$, $\alpha = \frac{1}{2\lambda}$, $\beta = \frac{1}{2\lambda}$. Подставим найденные значения параметров в (11) и получим семейство формул Рунге-Кутты второго порядка

$$y_{i+1} = y_i + h\left[(1-\lambda)f(x_i,y_i) + \lambda f(x_i + \frac{1}{2\lambda}h, y_i + \frac{1}{2\lambda}hf(x_i,y_i))\right].$$

В частности, при $\lambda = \frac{1}{2}$ получаем метод Хойна:

$$y_{i+1} = y_i + \frac{h}{2}\left[f(x_i,y_i) + f(x_i+h, y_i+hf(x_i,y_i))\right],$$

а при $\lambda = 1$ - метод средней точки:

$$y_{i+1} = y_i + hf(x_i + \frac{h}{2}, y_i + \frac{h}{2}f(x_i,y_i))].$$

Метод Рунге-Кутты четвертого порядка. Обычно, на практике применяется метод Рунге-Кутты четвертого порядка. По методу Рунге-Кутты четвертого порядка приближенные значения неизвестной функции y_i в точках x_i определяются последовательно из следующего равенств

$$y_{i+1} = y_i + \frac{h}{6}(k_1 + 2\,k_2 + 2\,k_3 + k_4),$$

где

$$\begin{aligned} k_1 &= f(x_i,y_i),\\ k_2 &= f(x_i+\tfrac{h}{2}, y_i+\tfrac{h}{2}k_1),\\ k_3 &= f(x_i+\tfrac{h}{2}, y_i+\tfrac{h}{2}k_2),\\ k_4 &= f(x_i+h, y_i+hk_3), \end{aligned}$$

$i = 0,1,...,n-1$.

Можно доказать, что глобальная погрешность метода Рунге-Кутты p-порядка равна $o(h^p)$, то есть $|y_i - y(x_i)| < Ch^p$, где C не зависит от h.

Метод Рунге-Кутты является одношаговым методом.

n.1 Метод Эйлера

Рассмотрим пример численного решения задачи Коши (1) методом Эйлера. Очистим переменные на всякий случай

In[1]:= $\mathbf{x} = .$
$\mathbf{y} = .$

и определим правую часть уравнения (1) в виде

In[2]:= $\mathbf{f[x\_, y\_] = y^2 - 5x^2}$

Out[2]= $y^2 - 5x^2$

Дифференциальное уравнение (1) с такой правой частью есть уравнение типа Риккати, оно не имеет решения в элементарных функциях.

Введем шаг следования узлов и начальные условия.

In[3]:= $\mathbf{h = 0.1;}$
$\mathbf{x_0 = -1;}$
$\mathbf{y_0 = -1;}$

Для сравнения найденного решения воспользуемся встроенной функцией **NDSolve** и получим численное решение задачи Коши (метод решения встроенной функцией неизвестен).

Будем искать решение на отрезке $[x_0, x_0 + \triangle]$. Зададим правый конец интервала

In[4]:= $\triangle = \mathbf{1};$

Следовательно $[x_0, x_0 + \triangle] = [-1, 0]$.

In[5]:= $\mathbf{Clear[y];}$
$\mathbf{eq = NDSolve[\{y'[x] == f[x, y[x]],\ y[x_0] == y_0\}, y[x],}$
$\mathbf{\{x, x_0, x_0 + \triangle\}][[1]]}$

Out[5]= $\{y[x] \to InterpolatingFunction[(-1, 0), <>][x]\}$

Решение получено в стандартном для программы Mathematica виде через итнерполяционную функцию. В круглых скобках указан интервал, на котором получено решение. Он совпадает с тем интервалам, который указан в команде. Если совпадения нет, то появится сообщение о невозможности решить уравнение на указанном интервале и будет приведен допустимый интервал. Тогда нужно скорректировать задачу - изменить первоначальный интервал (изменить $\triangle$) в соответствии с указанным в функции Out[5].

Введем решение команды *NDSolve* в виде функции:

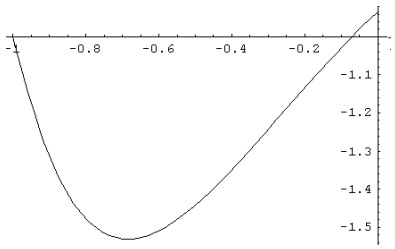
In[6]:= $\mathbf{y[x\_] = y[x]/.eq}$

Out[6]= $InterpolatingFunction[(-1, 0), <>][x]$

Теперь с такой функцией можно обращаться как обычно, например, построить ее график:

In[7]:= $\mathbf{gr = Plot[y[x], \{x, x_0, x_0 + \triangle\}, PlotStyle \to}$
$\mathbf{\{Thickness[0.01], Hue[0.7]\}];}$

Out[7]=



Сейчас обратимся к методу Эйлера. Введем число узлов без одного узла

In[8]:= $\mathbf{n} = \triangle / \mathbf{h}$

Out[8]= 10

Составим список, содержащий номер точки i, начиная со второй точки, саму точку x_i, значение численного решения y_i в данной точке x_i, найденное по формуле (9) и значение функции $y(x_i)$ в данной точке:

In[9]:= $\mathbf{p = Table[\{i, x_i = x_{i-1} + h,\ y_i = y_{i-1} + h\ f[x_{i-1}, y_{i-1}],\ y[x_i]\},\ \{i, 1, n\}];}$

Введем заголовок предыдущего списка

In[10]:= $\mathbf{q = \{"i","x_i","y_i","y[x_i]"\}}$

присоединим к списку начальное условие, значение функции $y[x]$ в начальной точке и напечатаем список

In[11]:= $\mathbf{p = Prepend[p, \{0, x_0, y_0, y[x_0]\}];}$
$\mathbf{(p = Prepend[p, q])//TableForm}$

Out[11]=

i	x_i	y_i	$y[x_i]$
0	-1	-1	-1
1	-0.9	-1.4	-1.31375
2	-0.8	-1.609	-1.47738
3	-0.7	-1.67011	-1.53055
4	-0.6	-1.63618	-1.50966
5	-0.5	-1.54847	-1.44254
6	-0.4	-1.4337	-1.34889
7	-0.3	-1.30815	-1.24243
8	-0.2	-1.18202	-1.13296
9	-0.1	-1.06231	-1.02793
10	0	-0.954456	-0.933559

Последние два столбца показывают отличие полученного решения от решения, полученного с помощью встроенной команды *DSolve*.

Ответом данной задачи считается список вида (x_i, y_i):

In[12]:= $\mathbf{p1 = Table[\{x_i, y_i\}, \{i, 0, n\}]}$

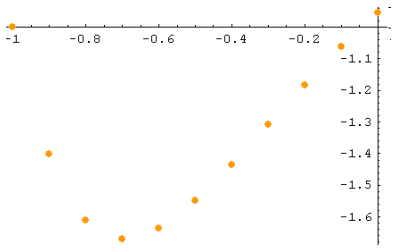
Out[12]=

{{ -1 , -1}, { -0.9 , -1.4}, { -0.8 , -1.609}, { -0.7 , -1.67011},
{ -0.6 , -1.63618}, { -0.5 , -1.54847}, { -0.4 , -1.4337}, { -0.3 , -1.30815},
{ -0.2 , -1.18202}, { -0.1 , -1.06231}, { 0 , -0.954456}}

Построим точки списка **p1** :

In[13]:= $\mathbf{gr1 = ListPlot[p1, PlotStyle \to \{Hue[0.1], PointSize[0.02]\}]}$

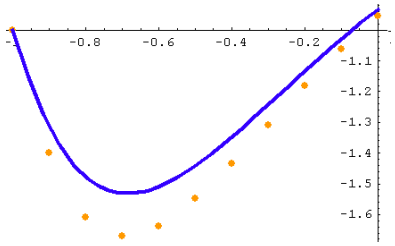
Out[13]=



и выведем их на общий график

In[14]:= **Show[gr, gr1];**

Out[14]=



n.2 Метод Рунге-Кутты

Решим задачу Коши предыдущего пункта методом Рунге-Кутты четвертого порядка. Очистим переменне:

In[1]:= **Clear[x, y, f, k1, k2, k3, k4]**

Введем функцию из (1)

In[2]:= **f[x_, y_] = y^2 − 5x^2**

Функция определена на том же интервале, интервал разбит так же точками, следующими с шагом $h = 0.1$.

Введем начальные условия, шаг следования точек, определим правый конец интервала и число узлов без одного узла.

In[3]:=
```
x0 = −1;
y0 = −1;
h = 0.1;
△= 1;
n =△ /h;
```

Введем заголовок таблицы с решениями

In[4]:= **q = {"i","xi","yi","y[xi]"}**

и составим цикл, в котором последовательно вычисляются величины k_i, значение табличной функции - решения в точке x_i и составляется строка таблицы ответа в виде списка $e[i]$. Цикл проработает n раз, тем самым рассмотрим все точки отрезка $[x_0, x_0 + \triangle]$:

In[5]:=
```
Do[k1 = f[xi, yi];
   k2 = f[xi + h/2, yi + (h/2) k1];
   k3 = f[xi + h/2, yi + (h/2) k2];
   k4 = f[xi + h, yi + h k3];
   yi+1 = yi + (h/6) (k1 + 2 k2 + 2 k3 + k4);
   e[i] = {i, xi, yi, y[xi]};
   xi+1 = xi + h, {i, 0, n}]
```

Составим таблицу и присоединим к ней заголовок

In[6]:=
```
q2 = Table[e[i], {i, 0, n}];
N[Join[{q}, q2], 6]//TableForm
```

Out[6]=

i	x_i	y_i	$y[x_i]$
0	-1	-1	-1
1	-0.9	-1.31375	-1.31375
2	-0.8	-1.47727	-1.47738
3	-0.7	-1.53043	-1.53055
4	-0.6	-1.50955	-1.50966
5	-0.5	-1.44245	-1.44254
6	-0.4	-1.34882	-1.34889
7	-0.3	-1.24238	-1.24243
8	-0.2	-1.13291	-1.13296
9	-0.1	-1.0279	-1.02793
10	0	-0.933533	-0.933559

Из последних двух столбцов видно, что найденное решение методом Рунге-Кутты достаточно хорошо совпадает с решением полученным с помощью встроенной командой *DSolve*.

Ответом данной задачи является список

In[7]:= **pr = Table[{x$_i$, y$_i$}, {i, 0, n}]**

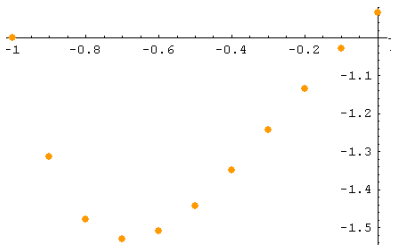
Out[7]=

{{ -1 , -1}, { -0.9 ,-1.31375}, { -0.8 , -1.47727 }, { -0.7 , -1.53043},
{ -0.6 , -1.50955}, { -0.5 , -1.44245 }, { -0.4 , -1.34882 }, { -0.3 , -1.24238},
{ -0.2 ,-1.13291 }, { -0.1 , -1.0279}, { 0 , -0.933533}}

Построим точки с координатами из списка *pr* :

In[8]:= **gr2 = ListPlot[pr, PlotStyle → {Hue[0.1], PointSize[0.02]}]**

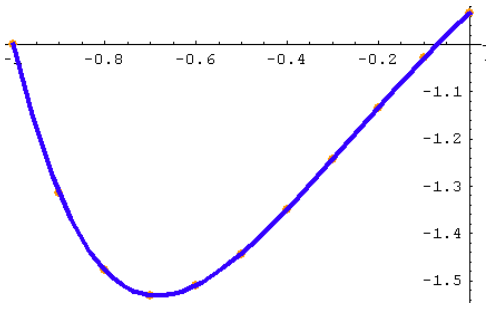
Out[8]=



Выведем графики.

In[11]:= **Show[gr, gr2];**

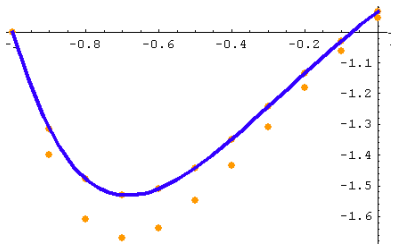
Out[11]=



В заключении приведем общий график, содержащий решения двух методов и решение данного дифференциального уравнения встроенной функцией.

In[12]:= **Show[gr, gr1, gr2];**

Out[12]=



По методу Рунге-Кутта четвертого порядка приходится четыре раза вычислять значения функции на каждом шаге, что может вызвать вычислительные трудности, если функция достаточно сложная.

Следующий метод Адамса требует знания значения функции только в одной точке начиная с некоторого шага.

n.3 Метод Адамса

Метод Адамса также решает задачу Коши (1) и является многошаговым методом. Рассмотрим идею этого метода. Разобьем отрезок $[a, b]$ точками $a = x_0 < x_1 < ... < x_n = b$ на n отрезков длины $h = (b - a)/n$ и проинтегрируем (1) от x_k до x_{k+1}:

$$y(x_{k+1}) = y(x_k) + \int_{x_k}^{x_{k+1}} f(x, y(x))dx \tag{15}$$

Допустим, что приближенные значения решения задачи Коши y_i уже найдены для первых $k + 1 < n$ точек, $i = 0, 1, ..., k$. Эту часть решения можно найти с помощью метода Эйлера или Рунге-Кутты. Обозначим через $f_i = f(x_i, y_i)$. По табличной функции (x_i, f_i), $i = 0, 1, ..., k$ построим многочлен Лагранжа k-го порядка $L_k(x)$. Многочлен Лагранжа будет аппроксимировать подинтегральную функцию (1) на интервале $[x_0, x_k]$. Заменим в (1) подитегральную функцию многочленом Лагранжа, а точные значения $y(x_i)$ на приближенные значения y_i:

$$y_{k+1} = y_k + \int_{x_k}^{x_{k+1}} L_k(x)dx \tag{16}$$

Получили приближенное решение y_{k+1}. Для определения y_{k+2} по табличной функции (x_i, f_i), $i = 1, ..., k+1$ построим другой многочлен Лагранжа k-го порядка $L_k(x)$ и воспользуемся формулой (16), сделав замену $k \to k+1$, и так далее. Число k фиксировано на каждом шаге и является степенью многочлена Лагранжа.

Пусть $m \geqslant k$. Возьмем многочлен Лагранжа $L_k(x)$ построенный по табличной функции (x_i, f_i), $i = m - k, m - k + 1, ..., m$,

$$L_k(x) = \sum_{j=m-k}^{m} f_j \prod_{\substack{i=m-k \\ i\neq j}}^{m} \frac{x - x_i}{x_j - x_i}.$$

Тогда

$$\int_{x_m}^{x_{m+1}} L_k(x)\,dx = \int_{x_m}^{x_{m+1}} \Big(\sum_{j=m-k}^{m} f_j \prod_{\substack{i=m-k \\ i\neq j}}^{m} \frac{x - x_i}{x_j - x_i} \Big)\,dx =$$

$$\sum_{j=m-k}^{m} f_j \int_{x_m}^{x_{m+1}} \prod_{\substack{i=m-k \\ i\neq j}}^{m} \frac{x-x_i}{x_j-x_i}\,dx = h \sum_{j=m-k}^{m} f_j\, c(k,m,j),$$

где

$$c(k,m,j) = \frac{1}{h}\int_{x_m}^{x_{m+1}} \prod_{\substack{i=m-k \\ i\neq j}}^{m} \frac{x-x_i}{x_j-x_i}\,dx. \tag{17}$$

Поэтому (16) можно переписать в следующем виде:

$$y_{m+1} = y_m + h \sum_{j=m-k}^{m} f_j\, c(k,m,j), \tag{18}$$

$m = k, k+1, ..., n-1$. При $k = 3$, например, формула (18) принимает вид (19).

Доказано, что если функция $f(x,y)$ $n+1$ раз непрерывно дифференцируема, то для приближенных значений $y_0, y_1, ..., y_n$, полученных по формуле (5), справедлива оценка

$$|y(x_i) - y_i| \leqslant Ch^{k+1},$$

$i = 0, 1, ..., n$, где $y(x)$ - аналитическое решение задачи Коши (1).

Сравнивая методы Рунге-Кутты четвертого порядка точности и метод Адамса при $k = 3$ так же четвертого порядка точности, видим, что на каждом шаге метод Адамса требует вычисления одного значения функции $f(x,y)$, а метод Рунге-Кутты - четырех вычислений значений функции. Но, метод Рунге-Кутты не требует знания нескольких начальных значений и позволяет менять шаг следования точек в любой момент вычисления.

Решим задачу п.1 методом Адамса. Введем функцию

In[1]:= $\mathbf{f[x\_, y\_] = y^2 - 5x^2}$

Функция определена на том же интервале, интервал разбит так же точками, следующими с шагом $h = 0.1$.

Введем начальные условия, шаг следования точек, определим правый конец интервала и число узлов без одного узла.

In[2]:= $\mathbf{x_0 = -1}$;
$\mathbf{y_0 = -1}$;
$\mathbf{h = 0.1}$;
$\mathbf{\triangle = 1}$;
$\mathbf{n = \triangle\ /h}$;

Введем узлы

In[3]:= $\mathbf{p = Table[x_{i+1} = x_i + h, \{i, 0, 9\}]//N}$;

Положим $k = 3$. Методом Рунге-Кутты найдем приближенные решения задачи Коши в первых четырех точках:

In[3]:= $\mathbf{Do[k_1 = f[x_i, y_i]}$;
$\mathbf{k_2 = f[x_i + \frac{h}{2}, y_i + \frac{h}{2}k_1]}$;
$\mathbf{k_3 = f[x_i + \frac{h}{2}, y_i + \frac{h}{2}k_2]}$;
$\mathbf{k_4 = f[x_i + h, y_i + k_3]}$;
$\mathbf{y_{i+1} = y_i + \frac{h}{6}\ (k_1 + 2\ k_2 + 2k_3 + k_4)}$;
$\mathbf{e[i] = \{i, x_i, y_i\}}$;
$\mathbf{x_{i+1} = x_i + h, \{i, 0, k\}]}$

Составим таблицу значений и присоединим к ней заголовок

In[4]:=
```
q = Table[e[i], {i, 0, k}];
Join[{"i", "x_i", "y_i"}, q]//TableForm
```

Out[4]=

i	x_i	y_i
0	-1	-1
1	-0.9	-1.31375
2	-0.8	-1.47727
3	-0.7	-1.53043

Введем функцию (17):

In[5]:=
```
c[k_, m_, j_] := 1/h Integrate[(Product[If[i == j, 1, (x - x_i)/(x_j - x_i)], {i, m - k, m}]), {x, x_m, x_(m+1)}]
```

Отметим, что функция $c(k, m, j)$ зависит от k, j, но не зависит от m. Действительно, возьмем, например, два значения $m = 3$ и $m = 6$ и напишем все значения функции, считая, что $k = 3$.

In[6]:=
```
m = 3; k = 3; Table[c[k, m, j], {j, m - k, m}]
```

Out[6]= $\{-\frac{3}{8}, \frac{37}{24}, -\frac{59}{24}, \frac{55}{24}\}$

In[7]:=
```
m = 6; k = 3; Table[c[k, m, j], {j, m - k, m}]
```

Out[7]= $\{-\frac{3}{8}, \frac{37}{24}, -\frac{59}{24}, \frac{55}{24}\}$

Поэтому формулу (18) при $k = 3$ можно написать в следующем виде

$$y_{m+1} = y_m + h(-\frac{3}{8}f_{m-3} + \frac{37}{24}f_{m-2} - \frac{59}{24}f_{m-1} + \frac{55}{24}f_m) \tag{19}$$

Теперь можно найти остальные значения приближенного решения $y_4, ..., y_{10}$. Учитывая, что $f_i = f(x_i, y_i)$, составим цикл:

In[8]:=
```
Do[y_(m+1) = y_m + h(-3/8 f[x_(m-3), y_(m-3)] + 37/24 f[x_(m-2), y_(m-2)] -
    59/24 f[x_(m-1), y_(m-1)] + 55/24 f[x_m, y_m]), {m, 3, 9}]
```

Таким образом, получили следующее решение задачи Коши методам Адамса:

In[9]:=
```
p = Table[{x_i, y_i}, {i, 0, 10}]//N;
```

Введем заголовок списка p

In[10]:=
```
q = {"x_i", "y_i"}
```

и присоединим его к списку p:

In[11]:=
```
Prepend[p, q]//TableForm
```

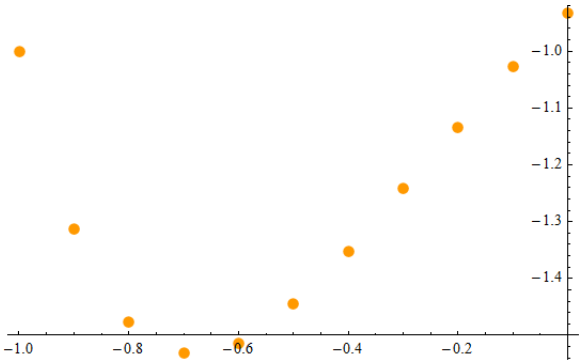
Out[11]=

x_i	y_i
$-1.$	$-1.$
-0.9	-1.31368
-0.8	-1.47727
-0.7	-1.53043
-0.6	-1.51328
-0.5	-1.44422
-0.4	-1.3516
-0.3	-1.24216
-0.2	-1.13365
-0.1	-1.02669
0	-0.93336

Построим график табличной функции p - решения задачи Коши:

In[12]:= **gr1 = ListPlot[p, PlotStyle → {Hue[0.1], PointSize[0.02]}]**

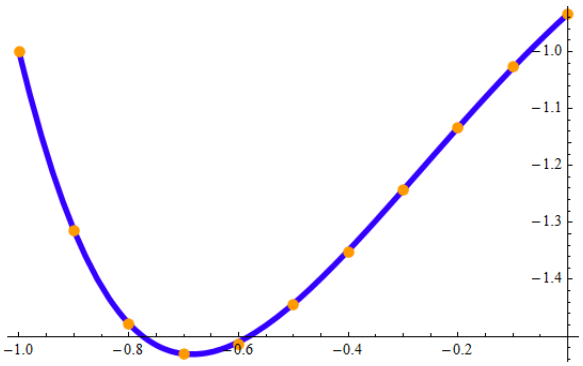
Out[11]=



и совместим его с графиком gr решения задачи Коши с помощью встроенной функции (см. п.1):

In[12]:= **Show[gr, gr1]**

Out[11]=



§21 Системы дифференциальных уравнений первого порядка

Известно, что к системе дифференциальных уравнений первого порядка сводится любое дифференциальное уравнение или система уравнений. Так, например, решение дифференциального уравнения

$$y'''(x) + (x^2+1)y''(x) - y'(x) + xy(x) - x^3 = 0,$$

введением новых неизвестных функций $u(x)$, $v(x)$ таких, что $y'(x) = u(x)$, $u'(x) = v(x)$, сводится к решению системе трех уравнений первого порядка относительно трех неизвестных функций.

$$\begin{cases} y'(x) = u(x), \\ u'(x) = v(x), \\ v'(x) = -(x^2+1)v(x) + u(x) - xy(x) + x^3. \end{cases}$$

Приведем два метода численного решения дифференциальных задач, содержащих такие системы.

Рассмотрим задачу Коши для системы дифференциальных уравнений первого порядка. Требуется найти функции $y_1(x), ..., y_n(x)$, определенные на отрезке $[a = x_0, b]$, удовлетворяющие системе дифференциальных уравнений

$$\begin{cases} y_1'(x) = f_1(x, y_1(x), ..., y_n(x)), \\ y_2'(x) = f_2(x, y_1(x), ..., y_n(x)), \\ \quad . \quad . \quad . \\ y_n'(x) = f_n(x, y_1(x), ..., y_n(x)) \end{cases} \tag{1}$$

и начальным условиям:

$$y_1(x_0) = y_1^0, \ \ldots, y_n(x_0) = y_n^0. \tag{2}$$

Запишем систему (1) в векторном виде. Введем векторы $Y(x) = (y_1(x), ..., y_n(x))$, $Y_0 = (y_1^0, ..., y_n^0)$, $Y'(x) = (y_1'(x), ..., y_n'(x))$. Функцию $f_i(x, y_1(x), ..., y_n(x))$ коротко запишем так $f_i(x, Y(x))$, $i = 1, 2, ..., n$, а вектор, состоящий из функций, стоящих в правых частях равенств (1), в виде $F(x, Y(x)) = (f_1(x, Y(x)), ..., f_n(x, Y(x))$.

Теперь задача Коши (1), (2) допускает следующую компактную векторную запись:

$$Y'(x) = F(x, Y(x)), \qquad Y(x_0) = Y_0. \tag{3}$$

Для численного решения задачи Коши (1), (2) или (3) надо задать натуральное число n, разбить отрезок $[a, b]$ точками $x_0 = a < x_1 < ... < x_n = b$ на n частей длины $h = (b-a)/n$ и найти табличную вектор-функцию (x_i, Y_i), $i = 1, 2, ..., n$, такую, что вектор $Y_i \approx Y(x_i)$, где $Y(x)$ вектор аналитического решения задачи Коши, который, вообще говоря, неизвестен.

На многомерный случай обобщаются методы решения одномерной задачи Коши, например, такие, как метод Эйдера или Рунге-Кутты.

По методу Эйлера решение задачи Коши (3) определяется по формуле

$$Y_{i+1} = Y_i + hF(x_i, Y_i), \quad i = 0, 1, ..., n-1. \tag{4}$$

По методу Рунге-Кутты решение задачи Коши (3) определяется по формуле

$$Y_{i+1} = Y_i + h(k_1 + 2k_2 + 2k_3 + k_4)/6, \quad i = 0, 1, ..., n-1. \tag{5}$$

где

$$k_1 = (f_1(x_i, Y_i), ..., f_n(x_i, Y_i)) = F(x_i, Y_i),$$
$$k_2 = (f_1(x_i + h/2, Y_i + h/2k_1), ..., f_n(x_i + h/2, Y_i + h/2k_1)) = F(x_i + h/2, Y_i + h/2k_1),$$
$$k_3 = (f_1(x_i + h/2, Y_i + h/2k_2), ..., f_n(x_i + h/2, Y_i + h/2k_2)) = F(x_i + h/2, Y_i + h/2k_2),$$
$$k_4 = (f_1(x_i + h, Y_i + hk_3), ..., f_n(x_i + h, Y_i + hk_3)) = F(x_i + h, Y_i + hk_3).$$

Рассмотрим пример. Решим систему уравнений

$$\begin{cases} y_1'(x) = y_2(x), \\ y_2'(x) = \frac{x^2}{6} y_2(x) - 3(x + y_3(x))y_4(x) + 6x, \\ y_3'(x) = y_1(x) + (x^2 - 3y_1(x))y_2(x) + y_4(x), \\ y_4'(x) = y_3(x) + x, \end{cases}$$

на промежутке $[1, 2]$ с шагом $h = 0.1$ и начальными условиями $y_1(1) = y_3(1) = y_4(1) = 1$, $y_2(1) = 3$.

Перейдем к векторной записи. Введем функцию в правой части (3). При этом не будем писать аргументы и уберем индексы - программа не допускает при определении функций переменные с индексами.

In[1]:=
```
F[x_, {y1_, y2_, y3_, y4_}] = {y2, x^2/6 y2 - 3(x + y3)y4 + 6x,
                               y1 + (x^2 - 3y1)y2 + y4, y3 + x}
```

Отметим, что фигурные скобки в определении функции позволяют подставлять вместо 4 аргументов по отдельности один 4-х мерный вектор.

Введем шаг и начальные условия:

In[2]:=
```
h = 0.1; x0 = 1; Y0 = {1, 3, 1, 1};
```

Метод Эйлера. В соответствии с формулой (4), составим цикл.

In[3]:=
```
Do[
    Yi+1 = Yi + h F[xi, Yi];
    e[i] = {i, xi, Yi}//Flatten;
    xi+1 = xi + h,
  {i, 0, 10}];
```

Решение напечатаем в виде таблицы.

In[4]:=
```
qz = {"i", "xi", "y1,i", "y2,i", "y3,i", "y4,i"};
q1 = Table[e[i], {i, 0, 10}], qz];
Prepend[q1, qz]//TableForm;
```

Out[4]=

i	x_i	$y_{1,i}$	$y_{2,i}$	$y_{3,i}$	$y_{4,i}$
0	1	1	3	1	1
1	1.1	1.3	3.05	0.6	1.2
2	1.2	1.605	3.15951	0.02955	1.37
3	1.3	1.92095	3.44999	−0.739284	1.49296
4	1.4	2.26595	4.07603	−1.80302	1.54903
5	1.5	2.67355	5.23647	−3.39345	1.50872
6	1.6	3.1972	7.18984	−5.99701	1.31938
7	1.7	3.91618	10.197	−10.601	0.879679
8	1.8	4.93588	14.0572	−19.1544	−0.0104173
9	1.9	6.3416	15.842	−34.9227	−1.74586
10	2.	7.9258	0.639235	−58.8833	−5.04813

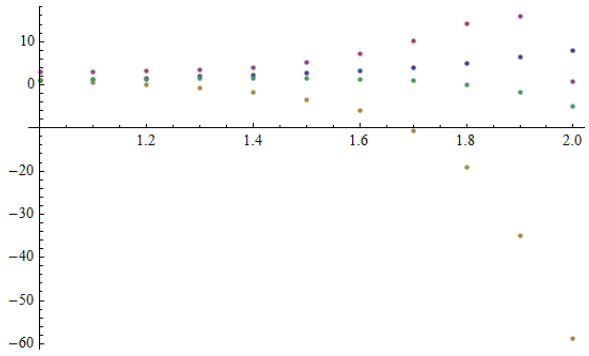
Построим график k-ой табличной функции $\{(x_i, y_{k,i})\}$, $i = 0, 1, ..., 10$, здесь $y_{k,i}$ есть i-ый элементы k-го столбца таблицы q1, $k = 3, ..., 6$. Следующая команда составляет список такой, что, каждый элемент списка есть матрица, у которой первый столбец состоит из координат узлов, а второй столбец есть k-ый элемент таблицы q1, $k = 3, ..., 6$, то есть предыдущей таблицы без заголовка.

In[5]:=
```
w = Table[q1[[1;;, {2, k}]], {k, 3, 6}];
```

Построим графики четырех табличных функций $(x_i, y_{k,i})$, $k = 1, 2, 3, 4$, $i = 0, 1, ..., 10$.

In[6]:=
```
tp1 = ListPlot[w, PlotRange → All, ]
```

Out[6]=



Для проверки найденного решения, сверим его с решением данной задачи Коши встроенной командой.

In[7]:=
```
Clear[y1, y2, y3, y4]
{y1[x_], y2[x_], y3[x_], y4[x_]} =
{y1[x], y2[x], y3[x], y4[x]} /.
NDSolve[{
    y1'[x] == y2[x], y2'[x] == x^2/6 y2[x] - 3 y4[x] (y3[x] + x) + 6 x,
    y3'[x] == y1[x] + y2[x] (x^2 - 3 y1[x]) + y4[x],
    y4'[x] == y3[x] + x, y1[1] == y3[1] == y4[1] == 1,
    y2[1] == 3,
    {y1[x], y2[x], y3[x], y4[x]},
{x, 1, 2}][[1]]
```

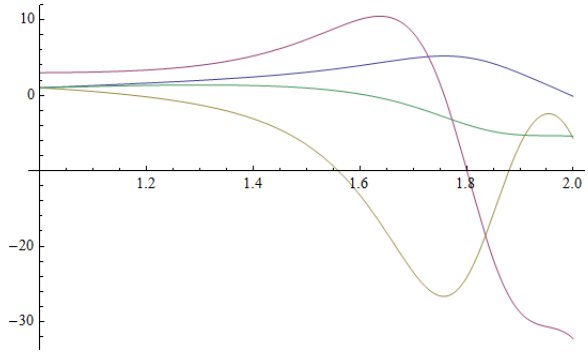
Out[7]=

$\{InterpolatingFunction[(1.\&2.), <>][x], InterpolatingFunction[(1.\&2.), <>][x],$
$InterpolatingFunction[(1.\&2.), <>][x], InterpolatingFunction[(1.\&2.), <>][x]\}$

Построим полученные решения.

In[8]:=
```
tp = Plot[{y1[x], y2[x], y3[x], y4[x]}, {x, 1, 2},
          PlotRange → All, AxesOrigin → {1, −10}]
```

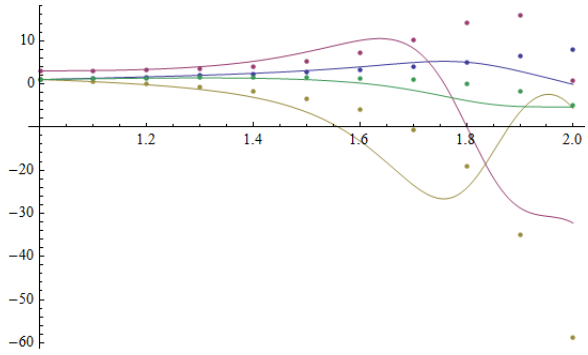
Out[9]=



Совместим теперь графики решений системы методом Эйлера и встроенной командой.

In[10]:=
```
Show[tp, tp1, AxesOrigin → {1, −10}]
```

Out[10]=



Расхождение полученных решений подчеркивает большую погрешность метода Эйлера.

Метод Рунге-Кутты. В соответствии с формулой (5), составим цикл.

In[3]:=
```
Do[
    k1 = F[xi, Yi];
    k2 = F[xi + h/2, Yi + h k1/2];
    k3 = F[xi + h/2, Yi + h k2/2];
    k4 = F[xi + h, Yi + h k3];
    Yi+1 = Yi + h(k1 + 2k2 + 2k3 + k4)/6;
    e[i] = {i, xi, Yi}//Flatten;
    xi+1 = xi + h,
  {i, 0, 10}];
qz = {"i", "xi", "y1,i", "y2,i", "y3,i", "y4,i"};
q2 = Table[e[i], {i, 0, 10}], qz];
```

In[4]:=
```
Prepend[q2, qz]//TableForm;
```

Out[4]=

i	x_i	$y_{1,i}$	$y_{2,i}$	$y_{3,i}$	$y_{4,i}$
0	1	1	3	1	1
1	1.1	1.30364	3.08984	0.51049	1.18208
2	1.2	1.62364	3.35285	−0.19915	1.31493
3	1.3	1.98599	3.97421	−1.27837	1.37031
4	1.4	2.43918	5.22176	−3.10287	1.29526
5	1.5	3.06163	7.38994	−6.53573	0.977888
6	1.6	3.93933	10.0452	−13.2258	0.180714
7	1.7	4.94075	8.20882	−23.4547	−1.4833
8	1.8	4.98188	−9.78427	−23.9672	−3.86305
9	1.9	2.91813	−28.3882	−6.63621	−5.20933
10	2.	−0.104494	−32.2498	−5.64809	−5.40955

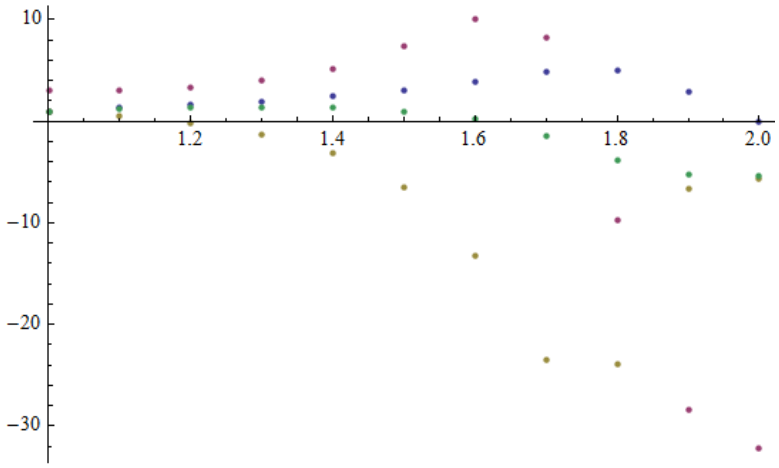
Составим список табличных функций, то есть, список решений.

In[5]:= **w = Table[q2[[1;;, {2, i}]], {i, 3, 6}];**

Построим все графики

In[6]:= **tp2 = ListPlot[w, PlotRange → All,]**

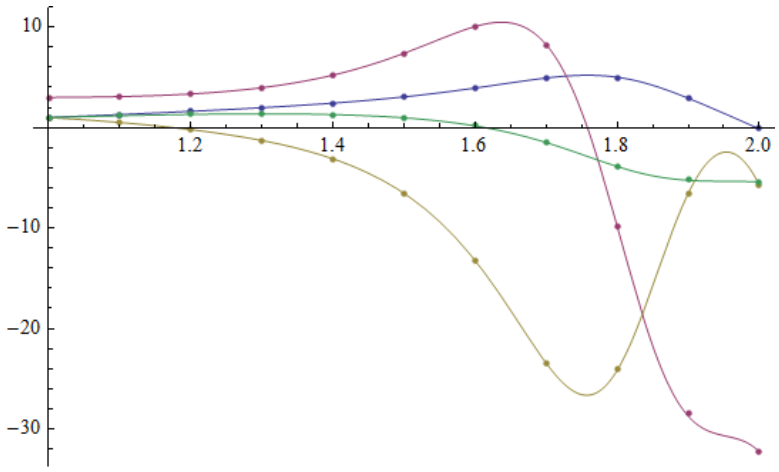
Out[6]=



Совместим графики данной решений задачи с помощью метода Рунге-Кутты и встроенной командой.

In[7]:= **Show[tp, tp2]**

Out[7]=



§22 Разностный метод. Аппроксимация, устойчивость, сходимость

Рассмотрим следующую краевую задачу: требуется найти на отрезке $[0,1]$ функцию $u(x)$ такую, что

$$\begin{array}{l} u''(x)+p(x)u'(x)-q(x)u(x)=f(x), \\ u(0)=\gamma_0, \quad u(1)=\gamma_1, \end{array} \tag{1}$$

$x \in [0,1]$, функции $p(x)$, $q(x) > 0$, $f(x)$ заданы на $[0,1]$; γ_0, γ_1 - заданные значения. Известно, что такая краевая задача имеет решение.

Для численного решения краевой задачи (1), зададим натуральное число n и разобьем отрезок $[0,1]$ на n равных отрезков точками (узлами) $x_0 = 0 < x_1 < ... < x_n = 1$. Длина каждого отрезка равна $h = (1-0)/n$. Такое множество точек называется сеткой (одномерной сеткой) на данном интервале. Узлы сетки x_0 и x_n называются граничными узлами, остальные узлы называются внутренними.

Функция, определенная на сетке, называется сеточной функцией. Для функции $\varphi(x)$, определенной на отрезке $[0,1]$, через $\widetilde{\varphi}$ обозначим сужение функции $\varphi(x)$ на множество узлов и получим сеточную функцию $\{\varphi(x_0), \varphi(x_1), ..., \varphi(x_n)\}$, при этом узлы не указываем и сеточная функция записана как вектор.

Введем норму сеточной функции, как максимальное по модулю значение сеточной функции.

Под численным решением краевой задачи (1) будем понимать сеточную функцию $y = \{y_0, y_1, ..., y_n\}$, такую, что y_i близки $u(x_i)$ для всех i, где $u(x)$ - аналитическое решение краевой задачи (1), как правило, неизвестное. Покажем как найти такое численное решение задачи (1) и поясним те требования, которые приводят к нужному численному решению задачи (1).

Решаем задачу (1) разностным методом. Запишем уравнение (1) для каждого внутреннего узла сетки и граничные условия:

$$\begin{array}{l} u''(x_i)+p(x_i)u'(x_i)-q(x_i)u(x_i)=f(x_i), \quad i=1,2,...,n-1 \\ u(0)=\gamma_0, \quad u(1)=\gamma_1, \end{array} \tag{2}$$

Получили систему уравнений относительно $u(x_i)$, $i = 0, 1, ..., n$, но такую систему не известно как решать, так как в системе присутствуют значения производных неизвестной функции. Преобразуем систему (2).

Выразим производные функции $u(x)$ во внутренних узлах через значения функции $u(x)$ в узлах по следующим формулам:

$$u'(x_i) = \frac{u(x_{i+1}) - u(x_{i-1})}{2h} - A_i h^2,$$

$$u''(x_i) = \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} - B_i h^2,$$

константы A_i, B_i содержат производные функции $u(x)$ в некоторых промежуточных точках (см. остаточный член ряда Тейлора в форме Лагранжа). Заменим производные в (2) по этим формулам, получим

$$m_i \equiv \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} + p_i \frac{u(x_{i+1}) - u(x_{i-1})}{2h} - q_i u(x_i) + Q_i h^2 = f_i, \tag{3}$$

$$u(0) = \gamma_0, \quad u(1) = \gamma_1, \tag{3'}$$

где $Q_i = -(A_i + B_i)$, $p_i = p(x_i)$, $q_i = q(x_i)$, $f_i = f(x_i)$, $i = 1, 2, ..., n-1$. Система (3), (3') называется разностной дифференциальной задачей.

Рассмотрим две сеточные функции, определенные на внутренних узлах: для решения $u(x)$ краевой задачи обозначим через $L\widetilde{u}$ сеточную функцию $(m_1, m_2, ..., m_{n-1})$, а через f сеточную функцию $(f(x_1), f(x_2), ..., f(x_{n-1})) = (f_1, f_2, ..., f_{n-1})$. Через $l\widetilde{u}$ обозначим сеточную функцию на граничных узлах $(u(0), u(1))$. Другими словами, оператор l ставит в соответствие сеточной функции её значения на граничных узлах. Пусть вектор $\gamma = (\gamma_0, \gamma_1)$. Тогда разностная дифференциальная задача может быть записана в виде

$$L\widetilde{u} = f, \quad l\widetilde{u} = \gamma. \tag{3*}$$

Действительно, такие равенства векторов записанные покоординатно приводят к системе (3), (3').

Рассмотрим еще одну систему (по аналогии с (3) (3')):

$$t_i \equiv \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + p_i \frac{y_{i+1} - y_{i-1}}{2h} - q_i y_i = f_i, \tag{4}$$

$$y_0 = \gamma_0, \quad y_n = \gamma_1, \tag{4'}$$

с $i = 1, 2, ..., n-1$. Это линейная система состоящая из $n+1$ уравнения относительно $n+1$ неизвестного $y_0, y_1, ..., y_n$. Система (4), (4') называется разностной схемой дифференциальной задачи (3), (3').

Пусть $y = (y_0, y_1, ..., y_n)$ сеточная функция решений системы (4), (4'). Через $L_h y$ обозначим сеточную функцию на внутренних узлах, состоящую из левых частей (4): $(t_1, t_2, ..., t_{n-1})$. Через $l_h y$ обозначим сеточную функцию на граничных узлах (y_0, y_n). Этот оператор совпадает с $l : l = l_h$. Тогда разностная схема (4), (4') может быть записана в виде

$$L_h y = f, \quad l_h y = \gamma. \tag{4*}$$

Несмотря на то, что уравнения систем (3), (3') и (4), (4') отличаются на малую величину порядка h^2 близость решений этих систем надо доказать. Только после этого решение $y = (y_0, y_1, ..., y_n)$ принимается в качестве численного решения краевой задачи (1).

Сначала покажем, что система (4), (4') имеет решение. Перепишем систему (4), (4') в следующем виде:

$$\begin{aligned} &y_0 = \gamma_0, \\ &y_{i-1}\Big(\frac{1}{h^2} - \frac{p_i}{2h}\Big) - y_i\Big(\frac{2}{h^2} + q_i\Big) + y_{i+1}\Big(\frac{1}{h^2} + \frac{p_i}{2h}\Big) = f_i, \\ &y_n = \gamma_1, \end{aligned}$$

$i = 1, 2, ..., n-1$. Так как матрица система трехдиагональная, то будем решать систему методом прогонки. Сначала проверим выполнение условия теоремы о корректности прогонки - является ли матрица системы матрицей с преобладанием диагональных элементов, то есть, верно ли следующее неравенство для всех i:

$$\Big|\frac{2}{h^2} + q_i\Big| > \Big|\frac{1}{h^2} - \frac{p_i}{2h}\Big| + \Big|\frac{1}{h^2} + \frac{p_i}{2h}\Big|.$$

Перепишем неравенство в виде:

$$|2 + h^2 q_i| > |1 - h\frac{p_i}{2}| + |1 + h\frac{p_i}{2}|.$$

Так как $q(x) > 0$, а h можно считать достаточно малым, то последнее неравенство превращается в верное неравенство:

$$2 + h^2 q_i > 1 - h\frac{p_i}{2} + 1 + h\frac{p_i}{2} = 2.$$

Таким образом система (4), (4') имеет решение.

Исследуем полученное решение $y = (y_0, y_1, ..., y_n)$.

Аппроксимация. Будем говорить, что разностная схема (4*) аппроксимирует дифференциальную задачу (3*) с k - ым порядком относительно h, если на решении $u(x)$ задачи (1) выполняется

$$||L\widetilde{u} - L_h\widetilde{u}|| \leqslant Dh^k,$$

где D - постоянная.

Обозначим через Q сеточную функцию $(Q_1, ..., Q_{n-1})$, определенную на внутренних узлах. Тогда на решении $u(x)$ для левые частей систем (3*), (4*) выполняется равенство

$$L\widetilde{u} = L_h\widetilde{u} + Qh^2. \tag{5}$$

Отсюда получаем, что

$$||L\widetilde{u} - L_h\widetilde{u}|| = ||Qh^2|| \leqslant Dh^2,$$

где $D = ||Q||$.

Следовательно разностная схема (4*) аппроксимирует дифференциальную задачу (3*) со 2 - ым порядком относительно h.

Устойчивость. Разностная схема (4*) устойчива, если для всех достаточно малых h и произвольных сеточных функций ξ и η разностная схема

$$L_h z = \xi, \quad l_h z = \eta \tag{6}$$

имеет единственное решение z и справедлива оценка

$$||z|| \leqslant C_1||\xi|| + C_2||\eta||, \tag{7}$$

где C_1, C_2 - постоянные, не зависящие от h, ξ и η.

Устойчивость означает, что малые изменения свободных членов системы влекут малые изменения решения и это свойство не зависит от размерности системы (то есть h). Действительно, пусть есть два решения этой системы полученные в результате изменения свободных членов (округления): $L_h z_1 = \xi_1$, $l_h z_1 = \eta_1$, $L_h z_2 = \xi_2$, $l_h z_2 = \eta_2$. Отсюда, вычитанием частей равенств, получим

$$L_h(z_1 - z_2) = \xi_1 - \xi_2, \quad l_h(z_1 - z_2) = \eta_1 - \eta_2.$$

Здесь мы предполагаем линейность операторов. Применим (7) к такой системе:

$$||z_1 - z_2|| \leqslant C_1||\xi_1 - \xi_2|| + C_2||\eta_1 - \eta_2||.$$

Отсюда получаем, что, если ξ_1 и ξ_2, η_1 и η_2 близки, то и z_1 и z_2 так же близки.

Можно доказать, что разностная схема (4*) устойчива (см. [4]).

Сходимость. Решение разностной схемы $y = (y_0, y_1, ..., y_n)$ сходится к решению $u(x)$ дифференциальной краевой задачи при $h \to 0$ с k - ым порядком относительно h, если

$$||y - \widetilde{u}|| < D_1 h^k,$$

D_1 - постоянная.

Разностная схема (4*) обладает свойством сходимости. Докажем это свойство. Разностная схема (4*) аппроксимирует дифференциальную задачу со 2-ым порядком относительно h, точнее для разностной схемы выполняется равенство (5). Поэтому, дифференциальную разностную задачу (3*) на решении задачи (1) можно записать в следующем виде

$$L_h \widetilde{u} + Qh^2 = f, \quad l\widetilde{u} = \gamma.$$

Для решения y разностной схемы выполняются равенства (4*):

$$L_h y = f, \quad l_h y = \gamma.$$

Так как $l = l_h$, то вычитая левые и правые части этих равенств, получим:

$$L_h(\widetilde{u} - y) = -Qh^2, \quad l_h(\widetilde{u} - y) = 0.$$

Так как разностная схема (4*) устойчива, то отсюда получим, что

$$||\widetilde{u} - y|| \leqslant C_1||-Qh^2|| + C_2||0|| \leqslant D_1 h^2,$$

где $D_1 = C_1||Q||$. Следовательно решение разностной схемы $y = (y_0, y_1, ..., y_n)$ сходится к решению $u(x)$ дифференциальной краевой задачи при $h \to 0$ со 2 - ым порядком относительно h.

§23 Разностный метод решения дифференциальных уравнений

Рассмотрим линейное дифференциальное уравнение второго порядка

$$p(x)\, y''(x) + q(x)\, y'(x) + r(x)\, y(x) + f(x) = 0, \tag{1}$$

где $p(x),\ q(x),\ r(x),\ f(x)$ функции определенные на отрезке $[a, b]$.

При поиске аналитического решения обычно рассматриваются две дифференциальные задачи.

Задача Коши. Найти решение $y(x)$ дифференциального уравнения (1) при условии, что $y(x_0) = t_0$, $y'(x_0) = t_1$, где $x_0 = a$, t_1, t_2 -заданные числа.

Краевая задача. Найти решение $y(x)$ дифференциального уравнение (1) при условии, что $y(a) = m_0$, $y(b) = m_1$, где m_0, m_1 -заданные числа.

Для численного решения каждой задачи перейдем от дифференциальной задачи к разностной схеме данной дифференциальной задачи (см. §22). Для этого зададим

натуральное число n и разобьем отрезок $[a,b]$ на n частей точками (узлами) $x_i = a + ih$, где $h = (b-a)/n$, $i = 0, 1, ..., n$. Будем искать сеточную функцию $y = \{y_0, y_1, ..., y_n\}$, определенную на данных узлах, являющуюся решением разностной схемы данной дифференциальной задачи. Для этого заменим в (1) первую и вторую производную центральной первой и второй разностной производной, то есть $(y_{i+1} - y_{i-1})/(2h)$ и $(y_{i-1} - 2y_i + y_{i+1})/h^2$ соответственно, а все остальные функции заменим их значениями в узлах. Здесь $y_0, y_1, ..., y_n$ пока неизвестные значения искомой сеточной функции. Получим разностное уравнение для каждого внутреннего узла:

$$\frac{y_{i-1} - 2y_i + y_{i+1}}{h^2} + q(x_i)\frac{y_{i+1} - y_{i-1}}{2h} + r(x_i)\ y_i + f(x_i) = 0 \qquad (2)$$

$i = 1, 2, ..., n-1$.

Теперь можно сформулировать численные задачи аналогичные двум приведенным дифференциальным задачам.

Численная задача Коши. Найти сеточную функцию $y = \{y_0, y_1, ..., y_n\}$, определенную на сетке $\{x_0, ..., x_n\}$ и удовлетворяющую системе уравнений (2) и начальным условиям $y_0 = t_0$, $(y_0 - y_1)/h = t_1$, где t_1, t_2 -заданные числа. Здесь $(y_0 - y_1)/h$ есть правая разностная производная табличной функции y в начальном узле x_0. Дать оценку абсолютной погрешности $|y_i - u(x_i)|$.

Численная краевая задача. Найти сеточную функцию $y = \{y_0, y_1, ..., y_n\}$, определенную на сетке $\{x_0, ..., x_n\}$ и удовлетворяющую системе уравнений (2) и граничным условиям условиям $y_0 = m_0$, $y_n = m_1$, где m_0, m_1 -заданные числа. Дать оценку абсолютной погрешности $|y_i - u(x_i)|$.

Система (2), добавленная начальными или граничными условиями, и есть разностная схема приведенной дифференциальной задачи.

При решении численной задачи Коши можно разрешить уравнение (2) относительно y_{i+1}, тем самым, выразим y_{i+1} через y_{i-1} и y_i. Так как y_0 определено первым начальным условием, а y_1 можно найти из второго начального условия, то из формулы (2) можно найти y_2, зная y_1 и y_2, по формуле (2) можно найти y_3 и так далее. В результате получим табличную функцию $y = \{y_0, y_1, ..., y_n\}$, $i = 1, 2, ..., n$, что и будет являться решением данной задачи Коши.

При решении численной краевой задачи, добавим к уравнениям (2) еще два уравнения $y_0 = m_0$, $y_n = m_1$ и полученную системы разрешим относительно y_i, $i = 0, 1, 2, ..., n$. Отметим, что и задачу Коши можно решить так же, добавив к разностным уравнениям (2) начальные условия $y_0 = t_0$, $(y_1 - y_0)/h = t_1$.

Если коэффициент при $y(x)$ в приведенном дифференциальном уравнении (1) будет отрицателен при все $x \in [a,b]$, то решение разностной схемы существует и сходится к аналитическому решению соответствующей дифференциальной задачи (см. §22) со вторым порядком относительно шага h : $|y_i - u(x_i)| \leqslant Ch^2$, где C - константа, не зависящая от h, $i = 1, 2, ..., n$.

В следующих двух пунктах приводятся решения следующей конкретной задачи. Найти численное решение дифференциального уравнения

$$y''(x) - \frac{y'(x)}{x} - 3\frac{y(x)}{x^2} - 6\ x = 0 \qquad (3)$$

на отрезке $[1, 2]$

а) с начальными условиями $y(1) = 1$, $y'(1) = 3$ (задача Коши).
б) с краевыми условиями $y(1) = 1$, $y(2) = 3$ (краевая задача).

n.1 Численная задача Коши.

Сформулируем численную задачу Коши для уравнения (2). Пусть $n = 10$. Разобьем отрезок $[1, 2]$ на $n = 10$ интервалов точками $x_i = 1 + 0.1i$ с шагом $h = 0.1$, $i = 0, 1, 2, ..., 10$. Уравнение (2) для данного дифференциального уравнения (3) примет следующий вид:

$$\frac{y_{i-1} - 2y_i + y_{i+1}}{h^2} - \frac{y_{i+1} - y_{i-1}}{2hx_i} - 3\,\frac{y_i}{x_i^2} + 6x_i = 0, \tag{4}$$

$i = 1, 2, ..., n - 1$, а начальные условия

$$y_0 = 1, \quad \frac{y_1 - y_0}{0.1} = 3. \tag{5}$$

Из второго начального условия находим $y_1 = 3\ 0.1 + 1 = 1.3$.

Уравнения (4) и (5) и есть разностная схема данной дифференциальной задачи Коши, ее решение и будет ответом. Как уже отмечалось, можно решить систему состоящую из уравнений (4) и (5). Но, можно получить более простую рекуррентную формулу. Для этого выразим y_{i+1} через y_{i-1} и y_i из уравнения (4). Поручим это вычисление и все остальные математическому пакету.

Введем следующие две команды для возможности повторно выполнить программу. Первая команда очищает переменную y, значением которой будет аналитическое решение задачи. Оно приводится только для сравнения с численным решением. Вторая команда очищает предыдущее численное решение задачи. Для первого раза необходимо задать значения таким переменным и тут же произвести очистку этим переменным - особенность индексированных переменных в Математике или выполнить команду Remove[y].

In[1]:=
```
Clear[y]
Do[y_i = 1; y_i = ., {i, 0, 10}]
```

Вводим шаг и левый конец интервала

In[2]:=
```
h = 0.1;
a = 1;
```

Перейдем к разностным уравнениям, совершив следующую подстановку:

In[3]:=
```
eq = (y''[x] - y'[x]/x - 3 y[x]/x^2 - 6 x) /. {y''[x] ->
  (y_(i-1) - 2y_i + y_(i+1))/h^2, y'[x] -> (y_(i+1) - y_(i-1))/(2h),
  y[x] -> y_i, x -> a + i h} // FullSimplify
```

Out[3]=

$$-6(0.1i + 1) + 100.\,(y_{i-1} - 2y_i + y_{i+1}) - \frac{5.\,(y_{i+1} - y_{i-1})}{0.1i + 1} - \frac{3y_i}{(0.1i + 1)^2}$$

Разрешим уравнение $eq = 0$ относительно y_{i+1}. Обычно команды 3 и 4 делают вручную, но Математика позволяет автоматизировать такие операции.

In[4]:=
```
v = Solve[eq == 0, y_(i+1)][[1, 1]] // FullSimplify
```

Out[4]=
$$(0.006(i + 9.99994)(i(i + 20.0001) + 100.001) - 1.(i + 10.)(i + 10.5)y_{i-1} + (i(2.i + 40.) + 203.)y_i)/((i + 9.5)(i + 10.))$$

Тем самым, найдено выражение y_{i+1} через y_i и y_{i-1}, что будет использовано в следующем цикле.

Запишем начальные условия. Отметим, что начальное условие с производной $y'(1) = 3$ нужно переписать, заменив производную правой разностной производной в точке $x_0 = 1$, и из полученного уравнения найти y_1 :

In[5]:=
```
y0 = 1;
x0 = 1;
y1 = 3 h + y0;
```

Произведем реккурентные вычисления табличной функции - решения задачи, используя выражение y_{i+1} через y_i и y_{i-1} :

In[6]:=
```
Do[yi+1 = (0.006 (i + 10)(i ( i + 20) + 100.001) - (i + 10)(i + 10.5)yi-1 +
      (i (2 i + 40) + 203)yi)/((i + 9.5)(i + 10)); xi = 1 + i h, {i, 1, 10}];
```

Для сравнения, решим данную задачу с помощью встроенной в Математику команды. В результате получим решение в элементарных функциях. Обычно таких решений задача не допускает.

In[7]:=
```
eqd = DSolve[{y''[x] - y'[x]/x - 3y[x]/x^2 - 6x == 0,
      y[1] == 1, y'[1] == 3}, y[x], x][[1]];
```

Определим функцию - решение дифференциальной задачи.

In[8]:=
```
y[x_] = y[x]/.eqd//FullSimplify
```

Out[8]= $(12\log(x)x^4 + 5x^4 + 3)/(8x)$

Введем строку - заголовок таблицы ответа:

In[9]:=
```
p = {"i","xi","yi","y[x]"};
```

Составим таблицу содержащую ответ и значения аналитического решения в узлах:

In[10]:=
```
p1 = Table[{i, xi, yi, y[xi]}, {i, 0, 10}];
```

и объединим ее с заголовком

In[11]:=
```
Join[{p}, p1]//TableForm
```

Out[11]=

i	x_i	y_i	$y[x_i]$
0	1	1	1
1	1.1	1.3	1.36307
2	1.2	1.73148	1.86508
3	1.3	2.31325	2.52621
4	1.4	3.0654	3.36778
5	1.5	4.00902	4.41204
6	1.6	5.16612	5.68208
7	1.7	6.55947	7.20168
8	1.8	8.2125	8.99529
9	1.9	10.1493	11.088
10	2.	12.3945	13.5053

Следующий список, собственно и является ответом задачи Коши:

In[12]:=
```
g = Table[{xi, yi}, {i, 0, 10}]
```

Out[12]=
```
{{1,1},{1.1,1.3},{1.2,1.73148},{1.3,2.31325},{1.4,3.0654},{1.5,4.00902},
 {1.6,5.16612},{1.7,6.55947},{1.8,8.2125},{1.9,10.1493},{2.,12.3945}}
```

Построим точки с координатами, указанными в списке g :

In[13]:=
```
s = ListPlot[g, PlotStyle → {PointSize[.02], Hue[.7]}];
```

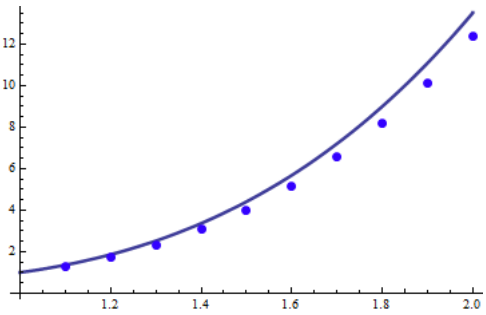
Построим аналитическое решение данной задачи Коши

In[14]:= **s1 = Plot[y[x], {x, 1, 2}, PlotStyle → {Thickness[0.007]}];**

и совместим построенные графики для сравнения:

In[15]:= **Show[s1, s]**

Out[15]=



n.2 Численная краевая задача.

Сформулируем численную краевую задачу для уравнения (2). Для этого добавим к уравнениям (4) предыдущего пункта граничные условия в следующем виде

$$y_0 = 1, \quad y_{10} = 3.$$

Получим систему $n+1$ уравнение относительно $n+1$ неизвестного $y_0, y_1, ..., y_n$. Решение данной системы будет и решением данной численной краевой задачи.

Введем команды для повторного выполнения программы

In[1]:= **Clear[y]**
Do[y$_i$ = 1; y$_i$ = ., {i, 0, 10}]

Зададим шаг следования узлов и левый конец интервала:

In[2]:= **h = 0.1; a = 1;**

Повторим команду (3) предыдущей задачи в несколько измененном виде

In[3]:= **eqi = (y''[x] − y'[x]/x − 3 y[x]/x^2 − 6 x == 0)/.{y''[x] →**
(y$_{i-1}$ − 2y$_i$ + y$_{i+1}$)/h^2, y'[x] → (y$_{i+1}$ − y$_{i-1}$)/(2h),
y[x] → y$_i$, x → a + i h})//FullSimplify

и получим разностные уравнения:

Out[3]= $1/(i+2)((i+2.)^3 - 1.3333\ (i+\ 2.5)\ y_{i-1}\ (i+2.) + 2.6666\ (i+\ 0.7752)\ (i+3.2247)\ y_i - 1.3333\ (i+1.5)\ (i+\ 1.9999)\ y_{i+1}) == 0$

Составим список из разностных уравнений для внутренних узлов:

In[4]:= **eq = Table[eqi, {i, 1, 9}]//FullSimplify**

Out[4]= $\{y_0 + 0.913043y_2 == 0.0631304 + 1.93676y_1,$
$y_1 + 0.92y_3 == 0.06912 + 1.94y_2,$
$y_2 + 0.925926y_4 == 0.0751111 + 1.94302y_3,$
$y_3 + 0.931034y_5 == 0.0811034 + 1.94581y_4,$
$y_4 + 0.935484y_6 == 0.0870968 + 1.94839y_5,$
$y_5 + 0.939394y_7 == 0.0930909 + 1.95076y_6,$
$y_6 + 0.942857y_8 == 0.0990857 + 1.95294y_7,$
$y_7 + 0.945946y_9 == 0.105081 + 1.95495y_8,$
$y_8 + 0.948718y_{10} == 0.111077 + 1.95682y_9$

Добавим к списку разностных уравнений граничные условия, записанные как уравнения:

In[5]:= $\mathbf{eq = Join[eq, \{y_0 == 1, y_{10} == 3\}]}$

Out[5]= $\{y_0 + 0.913043 y_2 == 0.0631304 + 1.93676 y_1,$
$y_1 + 0.92 y_3 == 0.06912 + 1.94 y_2,$
$y_2 + 0.925926 y_4 == 0.0751111 + 1.94302 y_3,$
$y_3 + 0.931034 y_5 == 0.0811034 + 1.94581 y_4,$
$y_4 + 0.935484 y_6 == 0.0870968 + 1.94839 y_5,$
$y_5 + 0.939394 y_7 == 0.0930909 + 1.95076 y_6,$
$y_6 + 0.942857 y_8 == 0.0990857 + 1.95294 y_7,$
$y_7 + 0.945946 y_9 == 0.105081 + 1.95495 y_8,$
$y_8 + 0.948718 y_{10} == 0.111077 + 1.95682 y_9,$
$y_0 == 1, y_{10} == 3\}$

Составим список из неизвестных y_i системы уравнений eq

In[6]:= $\mathbf{per = Table[y_i, \{i, 0, 10\}]}$

Out[6]= $\{y_0, y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y_9, y_{10}\}$

Разрешим систему уравнений eq относительно переменных per :

In[7]:= $\mathbf{sv = Solve[eq, per]}$

Out[7]= $\{\{y_0 \to 1., y_1 \to 0.772274, y_2 \to 0.612061, y_3 \to 0.526353,$
$y_4 \to 0.524626, y_5 \to 0.61821, y_6 \to 0.819877, y_7 \to 1.14357,$
$y_8 \to 1.6042, y_9 \to 2.21752, y_{10} \to 3\}\}$

Для сравнения найдем аналитическое решение краевой задачи, применяя встроенную функцию.

In[8]:= $\mathbf{eqd = DSolve[\{y''[x] - y'[x]/x - 3y[x]/x^2 - 6x == 0,}$
$\mathbf{y[1] == 1, y[2] == 3\}, y[x], x][[1]]//Simplify}$

In[9]:= $\mathbf{y[x\_] = y[x]/.eqd}$

Out[9]= $(20 + x^4(10 - 48\ Log[2]) + 48\ Log[2] + 45\ x^4\ Log[x])/(30x)$

Введем узлы решетки

In[10]:= $\mathbf{Do[x_i = 1 + i\ h, \{i, 0, 10\}]};$

заголовок таблицы

In[11]:= $\mathbf{p = \{"i", "x_i", "y_i", "y[x]"\}};$

Составим таблицу, состоящую из узлов, решения y_i системы уравнений sv и значений аналитического решения в узлах решетки:

In[12]:= $\mathbf{p1 = (Table[\{i, x_i, y_i, y[x_i]\}, \{i, 0, 10\}]/.sv)[[1]]};$

Добавим заголовок к таблице $p1$:

In[13]:= $\mathbf{Join[\{p\}, p1]//TableForm}$

Out[13]=

i	x_i	y_i	$y[x_i]$
0	1	1	1
1	1.1	0.772274	0.772102
2	1.2	0.612061	0.611916
3	1.3	0.526353	0.526329
4	1.4	0.524626	0.524752
5	1.5	0.61821	0.618474
6	1.6	0.819877	0.82024
7	1.7	1.14357	1.14397
8	1.8	1.6042	1.60456
9	1.9	2.21752	2.21775
10	2.	3.	3.

Составим список - ответ задачи

In[14]:= **gr = (Table[{x_i, y_i}, {i, 0, 10}]/.sv)[[1]]**

Out[14]= {{1,1},{1.1,0.772274},{1.2,0.612061},{1.3,0.526353},{1.4,0.524626},
{1.5,0.61821},{1.6,0.819877},{1.7,1.14357},{1.8,1.6042},{1.9,2.21752},{2,3}}

Построим точки с координатами в списке gr

In[15]:= **s = ListPlot[gr, PlotStyle → {PointSize[.02], Hue[1]}];**

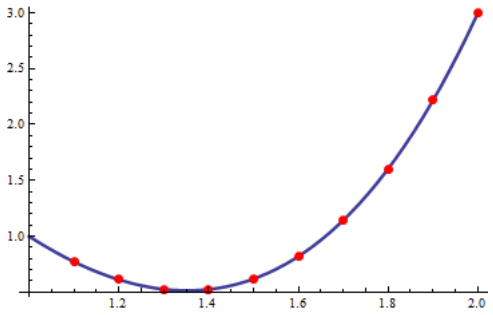
График аналитического решения

In[16]:= **s1 = Plot[y[x], {x, 1, 2}, PlotStyle → {Thickness[0.007]}];**

и выведем оба графика на экран:

In[17]:= **Show[s1, s]**

Out[17]=



§24 Задача Дирихле для уравнения Пуассона

Рассмотрим задачу Дирихле для уравнения Пуассона. Требуется найти решение $u(x, y)$ дифференциального уравнения второго порядка

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y) \tag{1}$$

в области $G = \{(x, y) | 1 \leqslant x, y \leqslant 2\}$, удовлетворяющее граничному условию

$$u(x, y)|_{\partial G} = g(x, y), \tag{2}$$

где ∂G - граница области G.

При численном решении данной дифференциальной задачи необходимо составить и решить разностную схему этой дифференциальной задачи.

Рассмотрим двумерную сетку в области G. Пусть число точек по ширине и длине сетки будет равно, например, 11. Тогда узлы сетки будут иметь координаты $(x_i, y_j) = (1 + h\,i, 1 + h\,j)$, $i, j = 0, 1, ..., 10$, где $h = (2 - 1)/10 = 0.1$ В сетке всего 121 узел. Узлы, не принадлежащие границе области ∂G называются внутренними, узлы, принадлежащие границе - граничными. Граничных узлов 40.

Будем искать сеточную функцию $\{u_{i,j}\}$, $i, j = 0, 1, ..., 10$, определенную на данной сетке, близкую к аналитическому решению дифференциальной задачи. Важно знать оценку погрешности найденного решения, то есть разностей $|u_{i,k} - u(x_i, u_k)|$.

Запишем уравнение (1) для каждого внутреннего узла сетки и заменим значения производных в узлах следующими разностными производными:

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i-1,k} - 2u_{i,k} + u_{i+1,k}}{h^2}, \quad \frac{\partial^2 u}{\partial y^2} \approx \frac{u_{i,k-1} - 2u_{i,k} + u_{i,k+1}}{h^2}.$$

После преобразований получим следующее уравнение в каждом внутреннем узле

$$u_{i-1,j} - 4u_{i,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} = h^2 f(x_i, y_j), \tag{3}$$

$i, j = 1, ..., 9$. Добавим к этим 81 уравнению 40 граничных условий для точек, принадлежащих ∂G :

$$u_{0,i} = g(1, y_i), \quad u_{10,i} = g(2, y_i), \quad u_{j,0} = g(x_j, 1), \quad u_{j,10} = g(x_j, 2) \tag{4}$$

$i = 0, 1, ..., 10$, $j = 1, ..., 9$.

Система состоящая из 121 уравнений (3) и (4) является разностной схемой данной дифференциальной задачи, решение которой $\{u_{i,j}\}$, $i, j = 0, 1, ..., 10$, требуется найти.

Известно, что построенная разностная схема (3), (4) имеет решение, которая сходится к решению исходной дифференциальной задачи со вторым порядком относительно шага h.

Рассмотрим пример численного решения задачи Дирихле для уравнения Пуассона.

Введем функции $f(x, y)$, $g(x, y)$, например, такие:

```
In[2]:= f[x_, y_] = 1/10  Sin[x y]
        g[x_, y_] = Sin[x] Sin[y]
Out[2]= 1/10 sin(x y)
Out[3]= sin(x) sin(y)
```

Пусть шаг следования узлов по обеим осям одинаков: $h = 0.1$ Введем координаты узлов:

```
In[4]:= Table[{x_i = 1 + i 0.1,  y_i = 1 + i 0.1}, {i, 0, 10}];
```

и составим разностное уравнение (3) для каждой внутренней точки (x_i, y_j)

```
In[5]:= eq1 = Table[u_{i-1,j} - 4u_{i,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} == f[x_i, y_j],
        {i, 1, 9}, {j, 1, 9}]//Flatten;
```

Добавим к этим уравнениям граничные условия (4):

```
In[6]:= eq2 = Table[{u_{0,i} == g[1, y_i], u_{10,i} == g[2, y_i]},
              {i, 0, 10}]//Flatten;
```

In[7]:= **eq3 = Table[{u$_{i,0}$ == g[x$_i$, 1], u$_{i,10}$ == g[x$_i$, 2]}, {i, 1, 9}]//Flatten;**

и объединим их с разносными уравнениями

In[8]:= **eq = Join[eq1, eq2, eq3];**

Введем список неизвестных значений функции $u_{i,j}$:

In[9]:= **per = Table[u$_{i,j}$, {i, 0, 10}, {j, 0, 10}]//Flatten;**

и решим систему относительно этих переменных

In[11]:= **s = Solve[eq, per];**

Выпишем список координат узлов и найденных решений разностной схемы для построения графика решения:

In[12]:= **gr = Table[{x$_i$, y$_j$, u$_{i,j}$}, {i, 0, 10}, {j, 0, 10}]/.s;**

Уберем внешние фигурные скобки

In[13]:= **sp = gr[[1]];**

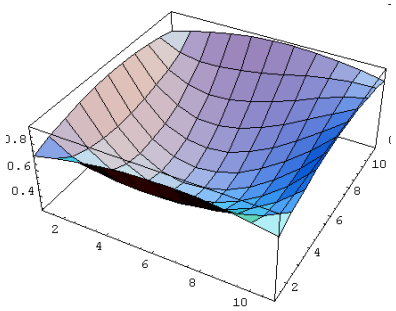
и лишние внутренние скобки

In[14]:= **sp = Flatten[sp, 1];**

и построим график решения

In[15]:= **ListPlot3D[gr]**

Out[14]=



§25 Уравнение колебания струны с закрепленными концами

Пусть концы плоской струны закреплены в точках $x = 0$ и $x = l$ оси x-ов.

Пусть график функции f(x), определенной на отрезке $[0, l]$, задает профиль плоской струны в начальный момент времени t=0, а функция $g(x)$ определяет начальную скорость точки струны с абсциссой x. Если функция $u[x, t]$, $x \in [0, l]$, описывает положение струны в момент времени $t \in [0, T]$, то $u[x, t]$ удовлетворяет дифференциальному уравнению:

$$\frac{\partial^2 u}{\partial t^2} = a^2 \frac{\partial^2 u}{\partial x^2} \tag{1}$$

начальным условиям:

$$u(x, 0) = f(x), \quad \frac{\partial u}{\partial t}(x, 0) = g(x) \tag{2}$$

и граничным условиям:

$$u(0,t)=0, \quad u(l,t)=0 \tag{2'}$$

Решение такой задачи можно получить приближенно, применяя разностный метод или точно, применяя метод Фурье.

Метод Фурье. По методу Фурье решение приведенной задачи Коши u[x,t] равно сумме бесконечного ряда Фурье:

$$u(x,t)=\sum_{k=1}^{\infty}(\alpha_k\cos\frac{ak\pi}{l}t+\beta_k\sin\frac{ak\pi}{l}t)\sin\frac{k\pi}{l}x \tag{3}$$

где

$$\alpha_k=\frac{2}{l}\int_0^l f(x)\sin(\frac{k\,\pi}{l}x)dx, \qquad \beta_k=\frac{2}{ak\pi}\int_0^l g(x)\sin(\frac{k\,\pi}{l}x)dx$$

Разностная схема. На прямоугольнике $[0,l]\times[0,T]$ рассмотрим сетку (x_i,t_k) состоящую из $(nh+1)\times(n\tau+1)$ точек, $x_i=0+h\,i$, $i=0,1,...,nh$, $t_k=0+\tau\,k$, $k=0,1,...,n\tau$, $h=l/nh$, $\tau=T/n\tau$.

Будем искать сеточную функцию $\{u_{i,j}\}$, $i=0,1,...,nh$, $k=0,1,...,n\tau$, определенную на данной сетке, близкую к аналитическому решению дифференциальной задачи (1) и (2), (2′). Важно знать оценку погрешности найденного решения, то есть разностей $|u_{i,k}-u(x_i,u_k)|$.

Запишем уравнение (1) в каждом внутреннем узле сетки и заменим значения вторых производных вторыми разностными производными.

$$\frac{\partial^2u}{\partial t^2}\approx\frac{u_{i,k-1}-2u_{i,k}+u_{i,k+1}}{\tau^2},$$

$$\frac{\partial^2u}{\partial x^2}\approx\frac{u_{i-1,k}-2u_{i,k}+u_{i+1,k}}{h^2}$$

Получим $nh-1$ разностное уравнение:

$$\frac{u_{i,k-1}-2u_{i,k}+u_{i,k+1}}{\tau^2}=a^2\frac{u_{i-1,k}-2u_{i,k}+u_{i+1,k}}{h^2} \tag{4}$$

Обозначим через

$$\gamma=a^2\frac{\tau^2}{h^2} \tag{5}$$

и выразим из уравнения (4) $u_{i,k+1}$:

$$u_{i,k+1}=-u_{i,k-1}+\gamma u_{i-1,k}+(2-2\gamma)u_{i,k}+\gamma u_{i+1,k} \tag{6}$$

Представим начальное условие (2) с производной приближенно в виде разностной производной

$$\frac{\partial u}{\partial t}(x_i,0)\approx\frac{u_{i,1}-u_{i,0}}{\tau} \tag{7}$$

Отсюда и из (2) находим $u_{i,1}$ и начальные условия (2) можно переписать так

$$u_{i,0} = f(x_i), \quad u_{i,1} = u_{i,0} + \tau\, g(x_i) \tag{8}$$

Начальные условия (8) и формула (6) позволяет найти все значения табличной функции - численного решения данной дифференциальной задачи $u_{i,k}$.

Доказано (см., например, [8]), что разностная схема устойчива, если выполняется соотношение шагов вида:

$$\tau \leqslant \frac{h}{a} \tag{9}$$

При выполнении этого условия разностное решение сходится к аналитическому решению, то есть

$$|u_{i,k} - u(x_i, u_k)| \to 0$$

при стремлении шагов к нулю $h \to 0$, $\tau \to 0$ с соблюдением условия (9).

Рассмотри конкретную задачу колебания струны и решим ее двумя способами.

Введем параметр струны и ее длину в состоянии покоя.

In[1]:= $\mathbf{a = 0.1; l = 5;}$

Пусть функция

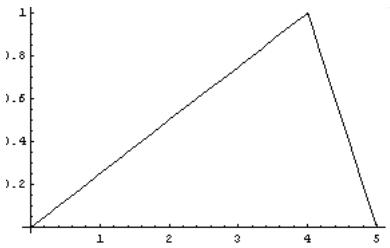
In[2]:= $\mathbf{f[x\_] = Which[x <= 4, (1/4)x, x > 4, -x + 5]}$

определенная на отрезке [0,5], задает профиль плоской струны в начальный момент времени t=0.

График этой функции:

In[3]:= $\mathbf{Plot[f[x], \{x, 0, 5\}]}$

Out[3]=



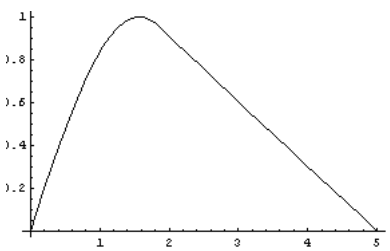
Пусть скорость точек струны при t=0 определяется функцией:

In[4]:= $\mathbf{g[x\_] = Which[x <= 2, Sin[x], x > 2, -Sin[2]/3(x - 5)]}$

Построим график функции скорости:

In[5]:= $\mathbf{Plot[g[x], \{x, 0, 5\}]}$

Out[5]=



Метод Фурье. Вводим функции, значения которых есть коэффициенты k-го члена ряда Фурье:

In[5]:=

$$\alpha[\mathbf{k\_}] = \frac{\mathbf{2}}{\mathbf{l}}\int_{\mathbf{0}}^{\mathbf{l}} \mathbf{f[x]Sin[}\frac{\mathbf{k}\,\pi}{\mathbf{l}}\mathbf{x]dx}$$

$$\beta[\mathbf{k\_}] = \frac{\mathbf{2}}{\mathbf{a\,k}\,\pi}\int_{\mathbf{0}}^{\mathbf{l}} \mathbf{g[x]Sin[}\frac{\mathbf{k}\,\pi}{\mathbf{l}}\mathbf{x]dx}$$

Out[5]= $$\frac{5(5\,sin[\frac{4\,k\,\pi)}{l}] - 4\,sin[k\,\pi])}{2k^2\pi^2}$$

Out[6]= $(-2273.24\,k\,cos[\frac{2\,k\,\pi}{5}] + (-1205.99 - 177.575\,k^2)sin[\frac{2\,k\,\pi}{5}] +$
$(1205.99 - 476.107\,k^2)sin[k\,\pi])/(-246.74\,k^3 + 97.4091\,k^5)$

Возьмем функцию u[x,t] приближенно - в виде суммы первых 4 членов ряда:

In[7]:=

$$\mathbf{u[x\_, t\_]} = \sum_{\mathbf{k=1}}^{\mathbf{4}}(\alpha[\mathbf{k}]\mathbf{Cos[}\frac{\mathbf{a\,k}\,\pi}{\mathbf{l}}\mathbf{t]} + \beta[\mathbf{k}]\mathbf{Sin[}\frac{\mathbf{a\,k}\,\pi}{\mathbf{l}}\mathbf{t]})\mathbf{Sin[}\frac{\mathbf{k}\pi}{\mathbf{l}}\mathbf{x]//N}$$

Out[7]=

```
(0.74443Cos[0.62831*t] + 13.5158*Sin[0.62831*t])* Sin[0.62831*x] +
(-0.301132*Sin[1.125663*t] + 2.23223*Cos[1.125663*t])*Sin[1.125663*x] +
(0.13383*Cos[0.188495*t] + 0.421293* Sin[0.188495*t])*Sin[1.88495*x] +
(-0.046527*Sin[0.251327*t] - 0.012378* Sin[0.251327*t])*Sin[2.51327*x])
```

Построим графики функции u[x,t] при t изменяющемся от 0 до 100 с шагом 1. Команда Epilog позволяет вместе с графиком выводить время t.

In[8]:=
```
Animate[Plot[u[x, t], {x, 0, l}, PlotStyle → {Hue[0.7]},
Epilog → {Text["t = " <> ToString[t], {5.5, 0.2}]},
PlotRange → {{0, l + 1}, {−18, 18}}], {t, 0, 100}]
```

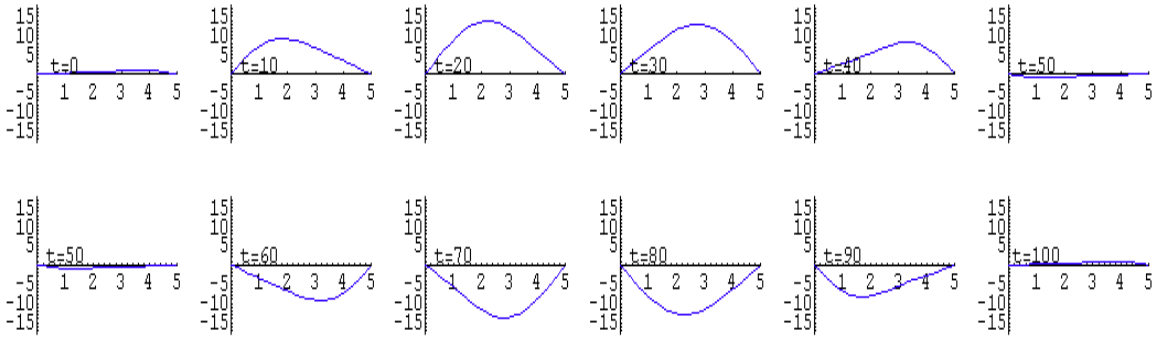
В результате получим анимацию 101 кадра - положений струны.

Приведем только несколько таких кадров

In[8]:=
```
gr1 = Table[Plot[u[x, t], {x, 0, 5}, PlotStyle → {Hue[0.7]},
Epilog → {Text["t = " <> ToString[t], 5.5, 0.2]},
PlotRange → {{0, 6}, {−18, 18}}], {t, 0, 50, 10}];
```

In[8]:=
```
gr2 = Table[Plot[u[x, t], {x, 0, 5}, PlotStyle → {Hue[0.7]},
Epilog → {Text["t = " <> ToString[t], {5.5, 0.2}]},
PlotRange → {{0, 6}, {−18, 18}}], {t, 50, 100, 10}];
```

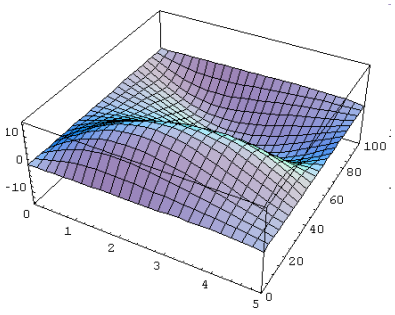
In[8]:=
```
Show[GraphicsArray[{gr1, gr2}]]
```



Наконец приведем все профили струны, для этого построим график функции u[x,t]

In[9]:= **Plot3D[u[x, t], {x, 0, 5}, {t, 0, 100}]**

Out[9]=



Разностный метод. Приведем численное решение задачи Коши для уравнения колебания струны. Вводим параметры задачи.

In[9]:= **l = 5;**
a = 0.1;
τ = 1;
h = 1/10;
nτ = 100/τ;
nh = l/h;
γ = a²τ²/h²;

Шаги выбраны так, что условие (9) выполняется. Рекомендуется вводить τ и h в виде рациональной дроби.

Определяем координаты узлов:

In[9]:= **Table[x_i = 0 + h i, {i, 0, nh}]//N;**
Table[t_i = 0 + τ i, {i, 0, nτ}]//N;

Записываем граничные и начальные условия (2) в виде (8):

In[9]:= **Table[u_{0,k} = u_{nh,k} = 0, {k, 0, nτ}]//N;**
Table[u_{i,0} = f[x_i]//N, {i, 0, nh}];
Table[u_{i,1} = u_{i,0} + τ g[x_i]//N, {i, 1, nh − 1}];

Вычисляем решения разностной схемы (6):

In[9]:= **Table[u_{i,k+1} = −u_{i,k−1} + γu_{i−1,k} + (2 − 2γ)u_{i,k} + γ u_{i+1,k},**
{k, 1, nτ − 1}, {i, 1, nh − 1}];

И записываем ответ разностной задачи:

In[9]:= **gr = Table[{x_i, u_{i,k}}, {k, 0, nτ}, {i, 0, nh}]**

Построим все найденные профили струны.

In[9]:= **Animate[ListPlot[gr[[k]], PlotJoined → True,**
PlotRange → {{0, 6}, {−15, 15}}], {k, 1, nτ}]

Получим анимацию 101 графика положений струны. Приведем только несколько профилей.

Составим два списка графиков выводимых профилей струны

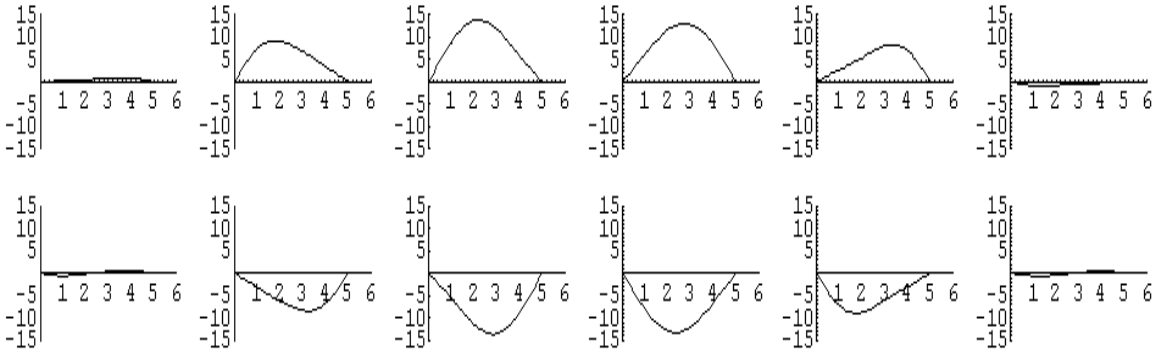
In[9]:= **gr1 = Table[ListPlot[gr[[k]], PlotJoined → True,**
PlotRange → {{0, 6}, {−15, 15}}], {k, 1, 60, 10}]

In[9]:= **gr2 = Table[ListPlot[gr[[k]], PlotJoined → True,**
PlotRange → {{0, 6}, {−15, 15}}], {k, 50, nτ + 1, 10}]

Выведем их на экран

In[9]:= **Show[GraphicsArray[{gr1, gr2}]]**

Out[9]=

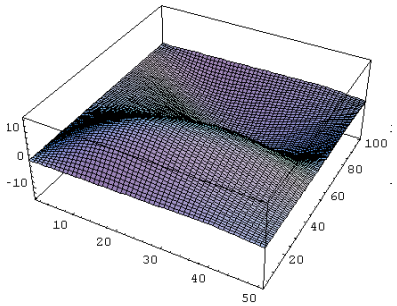


Теперь выведем все профили струны в виде трехмерной поверхности Для этого составим список значений функции

In[9]:= **grz = Table[$u_{i,k}$, {k, 0, nτ}, {i, 0, nh}]**

In[9]:= **ListPlot3D[grz]**

Out[9]=



Если выполнить оба решения последовательно, то будет найдены два решения - приближенное аналитическое решение $u[x,t]$ и численное решение $u[x_i,t_k]$, $i = 0,1,...,nh$, $k = 0,1,...,n\tau$. Для сравнения полученных решений можно посмотреть на графики профилей или сравнить трехмерные графики всех профилей.

Приведем так же максимальное значение модуля разности этих функций в узлах сетки:

In[9]:= **Max[Table[Abs[$u_{i,k}$ − u[0 + i h, 0 + k t]],**
{k, 0, nτ}, {i, 0, nh}]//Flatten]

Out[9]= 0.1364

§26 Уравнение теплопроводности

Обозначим через $u(x,y,t)$ температуру в точке (x,y) плоской изотропной квадратной пластины $\Omega = [0,1] \times [0,1]$ в момент времени $t \in [0,T]$.

Пусть начальное распределении температуры:

$$u(x, y, 0) = \varphi(x, y), \tag{1}$$

$(x, y) \in \Omega$, и для всех значений температуры $t \in [0, T]$, для всех граничных точек пластины $(x, y) \in \partial\Omega$ известно, что

$$u(x, y, t) = \psi(x, y, t). \tag{2}$$

При выполнении этих условий, функция $u(x, y, t)$ удовлетворяет уравнению вида

$$\frac{\partial u}{\partial t} = a^2\Big(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\Big) + f(x, y, t), \tag{3}$$

где a- параметр пластины.

Требуется найти решение $u(x, y, t)$ дифференциального уравнения (3), удовлетворяющее начально-граничным условиям (1) и (2).

Решаем задачу разностным методом. Рассмотрим три разностные схемы - явную, неявную и продольно-поперечную схему (или метод переменных направлений).

Будем решать конкретную задачу. Введем функции и значение параметра a :

In[1]:=
```
f[x_, y_, t_] = x^2 + y t;
φ[x_, y_] = 1;
ψ[x_, y_, t_] = E^(-x y t);
a = 1/10;
```

Рассмотрим сетку в параллелепипеде $D = [0, 1] \times [0, 1] \times [0, T]$. Пусть $n+1$ - число узлов сетки вдоль осей x и y, $m + 1$ - число узлов сетки вдоль оси времени t, h - шаг следования узлов вдоль осей x и y, t-шаг следования узлов по временной оси t. Введем также коэффициент γ разностных уравнений.

In[2]:=
```
T = 1;
n = 10;
m = 10;
h = 1/n;
t = T/m;
γ = a^2 t/h^2;
```

n.1 Явная разностная схема

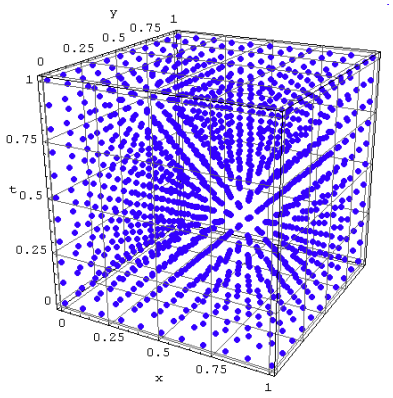
Введем координаты узлов:

In[3]:=
```
Table[{x_i = i h, y_i = i h}, {i, 0, n}];
Table[t_i = i t, {i, 0, m}];
```

и для наглядности нарисуем сетку следующими командами:

In[4]:=
```
q = Flatten[Table[{x_i, y_j, t_k}, {i, 0, n}, {j, 0, n}, {k, 0, m}], 2];
Show[Graphics3D[{Hue[0.7], PointSize[0.02],
Point /@ q}], ViewPoint → {1.501, -2.771, 1.231},
Axes → True, AxesLabel → {"x", "y", "t"},
FaceGrids → All]
```

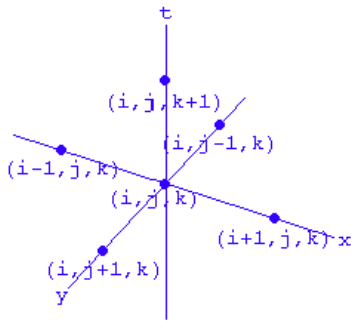
Out[4]=



Пусть сеточная функция $u_{i,j,k} \approx u(x_i, y_j, t_k)$. Тогда начально - граничные условия (1) и (2) с помощью такой сеточной функции можно записать в следующем виде:

In[5]:=
```
Table[u_{i,j,0} = φ[x_i, y_j], {i, 0, n}, {j, 0, n}];
Table[u_{0,i,k} = ψ[0, y_i, t_k], {i, 0, n}, {k, 0, m}];
Table[u_{n,i,k} = ψ[1, y_i, t_k], {i, 0, n}, {k, 0, m}];
Table[u_{i,0,k} = ψ[x_i, 0, t_k], {i, 1, n - 1}, {k, 0, m}];
Table[u_{i,n,k} = ψ[x_i, 1, t_k], {i, 1, n - 1}, {k, 0, m}];
```

Запишем дифференциальное уравнение (3) в виде разностных уравнений. Для этого заменим производные в (3) разностными производными.



Разностные производные в точке с координатами (x_i, y_j, z_k) определяется с помощью соседних точек, отмеченных на рисунке набором соответствующих индексов координат, по следующим формулам:

$$\frac{\partial u}{\partial t} \approx \frac{u_{i,j,k+1} - u_{i,j,k}}{t},$$

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i-1,j,k} - 2u_{i,j,k} + u_{i+1,j,k}}{h^2},$$

$$\frac{\partial^2 u}{\partial y^2} \approx \frac{u_{i,j-1,k} - 2u_{i,j,k} + u_{i,j+1,k}}{h^2},$$

Заменим производные в (3) соответствующими разностными производными и после преобразований получим следующие разностные уравнения:

$$u_{i,j,k+1} = u_{i,j,k} + \gamma(u_{i-1,j,k} + u_{i+1,j,k} + u_{i,j-1,k} + u_{i,j+1,k} - 4u_{i,j,k}) + t\ f[x_i, y_j, t_k],$$

где $\gamma = a^2 \frac{t}{h^2}$.

Разностные уравнения позволяют найти значения сеточной функции $u_{i,j,k+1}$ на $k+1$ слое, если известны все значения функции $u_{i,j,k}$ на ниже лежащем k слое.

Так как на основании параллелепипеда D и его боковых гранях значения функции $u_{i,j,k}$ заданы начально-граничными условиями, то значения функции в остальных точках сетки можно получить в результате выполнения следующего цикла:

```
In[6]:= Do[u_{i,j,k+1} = u_{i,j,k} + γ(u_{i-1,j,k} + u_{i+1,j,k} + u_{i,j-1,k} + u_{i,j+1,k} -
        4u_{i,j,k}) + t f[x_i, y_j, t_k], {k, 0, m - 1}, {i, 1, n - 1}, {j, 1, n - 1}]
```

Теперь ответ задачи можно записать в виде:

```
In[7]:= Flatten[Table[{x_i, y_j, t_k, u_{i,j,k}}, {i, 0, n}, {j, 0, n}, {k, 0, m}], 2];
```

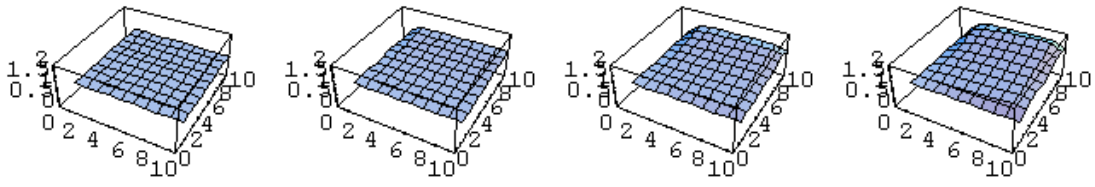
Для визуального восприятия решения введем список значений температур на k-ом слое:

```
In[8]:= g[k_] := Table[u_{i,j,k}, {i, 0, n}, {j, 0, n}];
```

и построим приближенно графики функций $u = u(x, y, t)$ для значений времени $t = 0, 1, ..., m$, идущего с шагом 3:

```
In[9]:= p = Table[ListPlot3D[g[k],
        PlotRange → {{0, n}, {0, n}, {0, 6}}], {k, 0, m, 3}];
```

Out[9]=



Рассмотренные разностные уравнение вместе с начальными условиями, введенными командой 5, есть **явная разностная схема**. Она аппроксимирует дифференциальную задачу с точностью $0(h^2 + t)$. Разностная схема устойчива при

$$t < \frac{h^2}{4a^2} \qquad (4).$$

Это неравенство накладывает существенные ограничения на шаг по времени, что может привести к большим вычислительным затратам при больших значениях T.

n.2 Неявная разностная схема

Условие (4) можно снять. Для этого надо от уравнения (3) перейти к разностным уравнениям, заменяя первую частную производную в (3) на левую разностную производную по формуле:

$$\frac{\partial u}{\partial t} \approx \frac{u_{i,j,k} - u_{i,j,k-1}}{t},$$

а вторые частные производные заменяются так же, как и в предыдущем случае. После подстановки разностных производных в (3) и преобразования, получим следующее разностное уравнение (**неявную разностную схему**):

$$\gamma u_{i-1,j,k} + \gamma u_{i+1,j,k} + \gamma u_{i,j-1,k} + \gamma u_{i,j+1,k} - (1 + 4\gamma) u_{i,j,k} =$$

$$-u_{i,j,k-1} - tf[x_i, y_j, t_k]$$

Запишем это разностное уравнение для каждого узла сетки параллелепипеда D, не принадлежащего нижнему основанию и боковым граням параллелепипеда[1]:

```
In[4]:= eq = Flatten[Table[γ u_{i-1,j,k} + γ u_{i+1,j,k} + γ u_{i,j-1,k} + γ u_{i,j+1,k} -
          (1 + 4γ) u_{i,j,k} + u_{i,j,k-1} == -(t f[x_i, y_j, t_k]//N) ,
          {k, 1, m}, {j, 1, n - 1}, {i, 1, n - 1}]];
```

Гранично-начальные условия запишем в виде следующих уравнений:

```
In[5]:= eq1 = Flatten[Table[u_{i,j,0} == φ[x_i, y_j], {i, 0, n}, {j, 0, n}]];
        eq2 = Flatten[Table[u_{0,i,k} == ψ[0, y_i, t_k], {i, 0, n}, {k, 1, m}]];
        eq3 = Flatten[Table[u_{n,i,k} == ψ[1, y_i, t_k], {i, 0, n}, {k, 1, m}]];
        eq4 = Flatten[Table[u_{i,0,k} == ψ[x_i, 0, t_k], {i, 1, n - 1}, {k, 1, m}]];
        eq5 = Flatten[Table[u_{i,n,k} == ψ[x_i, 1, t_k], {i, 1, n - 1}, {k, 1, m}]];
```

Объединим все уравнения в одну систему уравнений.

```
In[6]:= eq = Join[eq, eq1, eq2, eq3, eq4, eq5];
```

Получим систему из $(n+1)^2(m+1)$ уравнений и столько же неизвестных. Матрица системы eq является матрицей с преобладанием диагональных элементов и поэтому система однозначно разрешима.

Введем список переменных системы:

```
In[7]:= per = Flatten[Table[u_{i,j,k}, {k, 0, m}, {j, 0, n}, {i, 0, n}]];
```

Решаем систему встроенной функцией:

```
In[8]:= sw = Solve[eq, per][[1]];
```

Находим значения сеточной функции, то есть ответ поставленной задачи:

```
In[9]:= Do[u_{i,j,k} = u_{i,j,k}/.sw, {k, 0, m}, {j, 0, n}, {i, 0, n}];
```

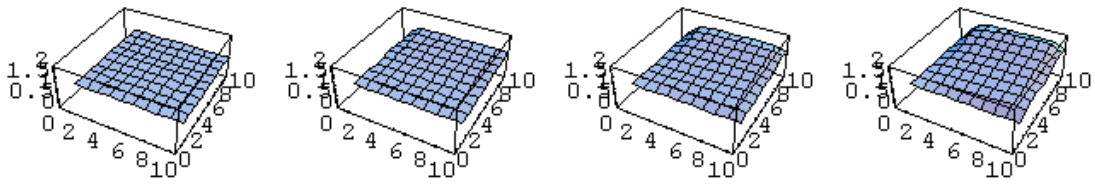
Для наглядности введем список значений температур на k-ом слое:

```
In[10]:= g[k_] := Table[u_{i,j,k}, {j, 0, 10}, {i, 0, 10}]
```

и построим приближенно графики функций $u = u(x, y, t)$ для значений времени $t = 0, 1, ..., m$, идущего с шагом 3:

```
In[11]:= p = Table[ListPlot3D[g[k], PlotRange →
          {{0, n}, {0, n}, {0, 2}}], {k, 0, m, 3}];
```

Out[11]=



К недостаткам данного подхода надо отнести большой объем вычислений при решении системы при малых шагах и большом интервале времени.

Есть более экономичные схемы решения задачи теплопроводности, позволяющие найти значение температуры в узлах одного слоя, производя, при этом, арифметических действий в количестве, имеющем один порядок с количеством узлов в слое.

Рассмотрим одну из таких схем.

[1] Первые три вводные команды выполнены выше.

n.3 Метод переменных направлений

Сетка D дополняется промежуточными временными слоями точками $u_{i,j,k+\frac{1}{2}} = u(x_i, y_j, z_{k+\frac{1}{2}})$, $k = 0, \frac{1}{2}, 1, \frac{3}{2}, ..., m$. Изменяются и гранично-начальные условия[1]:

```
In[3]:= Table[{xi = i h, yi = i h}, {i, 0, n}];
        Table[{ti = i t}, {i, 0, m, 1/2}];
        eq1 = Table[u_{i,j,0} = φ[xi, yj], {i, 0, n}, {j, 0, n}];
        eq2 = Table[u_{0,i,k} = ψ[0, yi, tk], {i, 0, n}, {k, 1/2, m, 1/2}];
        eq3 = Table[u_{n,i,k} = ψ[1, yi, tk], {i, 0, n}, {k, 1/2, m, 1/2}];
        eq4 = Table[u_{i,0,k} = ψ[xi, 0, tk], {i, 1, n - 1}, {k, 1/2, m, 1/2}];
        eq5 = Table[u_{i,n,k} = ψ[xi, 1, tk], {i, 1, n - 1}, {k, 1/2, m, 1/2}];
```

Первые две команды уже введены выше.

Во внутренних узлах промежуточного слоя дифференциальное уравнение (3) заменяется разностными, после следующей замены производных:

$$\frac{\partial u}{\partial t} \approx \frac{u_{i,j,k+\frac{1}{2}} - u_{i,j,k}}{t},$$

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i-1,j,k+\frac{1}{2}} - 2u_{i,j,k+\frac{1}{2}} + u_{i+1,j,k+\frac{1}{2}}}{h^2},$$

$$\frac{\partial^2 u}{\partial y^2} \approx \frac{u_{i,j-1,k} - 2u_{i,j,k} + u_{i,j+1,k}}{h^2},$$

После преобразований получим разностные уравнения:

$$u_{i-1,j,k+\frac{1}{2}} - (2 + \frac{2}{\gamma})u_{i,j,k+\frac{1}{2}} + u_{i+1,j,k+\frac{1}{2}} = (2 - \frac{2}{\gamma})u_{i,j,k} - u_{i,j-1,k} - u_{i,j+1,k} + \frac{t}{\gamma} f[x_i, y_j, t_{k+\frac{1}{2}}], \quad (5)$$

где $\gamma = a^2 \frac{t}{h^2}$.

При фиксированном k и фиксированном j получим систему из $n-1$ разностного уравнения с $n-1$ неизвестной (две неизвестные определены из начальных условий). Это трехдиагональная система с преобладанием диагональных элементов. Такую систему можно решить методом прогонки. Для всех k и j система имеет обну и ту же матрицу системы. Введем эту матрицу. Возьмем матрицу a первоначально с нулевыми элементами и заменим в ней соответствующие элементы в соответствии с коэффициентами разностного уравнения (5).

```
In[4]:= a = Table[0, {k, 1, n - 1}, {i, 1, n - 1}];
         a[[1, 1]] = -(2 + 2/γ);
         a[[1, 2]] = 1;
         Do[
              a[[i, i]] = -(2 + 2/γ);
              a[[i, i - 1]] = 1;
              a[[i, i + 1]] = 1,
         {i, 2, n - 2}];
         a[[n - 1, n - 1]] = -(2 + 2/γ);
```

[1] Первые две вводные команды выполнены в начале параграфа выше.

```
a[[n - 1, n - 2]] = 1;
```

Проверим результат:

In[5]:=
```
a//MatrixForm
```

Out[5]=

$$\begin{pmatrix} -22. & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -22. & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -22. & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -22. & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -22. & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -22. & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -22. & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -22. & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -22. \end{pmatrix}$$

Введем команду вычисляющую свободные члены системы (5):

In[6]:=
```
bkh[j_, k_] := Block[{b = Table[0, {n - 1}]},
  b[[1]] = -u[0, j, k + 1/2] - u[1, j + 1, k] - u[1, j - 1, k] +
  (2 - 2/γ) u[1, j, k] - t/γ f[x[1], y[j], t[k + 1/2]];
  b[[n - 1]] = -u[n, j, k + 1/2] - u[n - 1, j + 1, k] - u[n - 1, j - 1, k] +
  (2 - 2/γ) u[n - 1, j, k] - t/γ f[x[n - 1], y[j], t[k + 1/2]];
  Do[b[[i]] = -u[i, j + 1, k] - u[i, j - 1, k] + (2 - 2/γ) u[i, j, k] -
      t/γ f[x[i], y[j], t[k + 1/2]],
    {i, 2, n - 2}];
  b]
```

Следующая команда вычисляет прогоночные коэффициенты по матрице a и столбцу свободных членов $bkh[j,k]$:

In[7]:=
```
prog[j_, k_] := Block[{},
  d[1] = -a[[1, 2]]/a[[1, 1]];
  l[1] = bkh[j, k][[1]]/a[[1, 1]];
  Do[d[s] = -a[[s, s + 1]]/(a[[s, s - 1]] d[s - 1] + a[[s, s]]);
    l[s] = (bkh[j, k][[s]] - a[[s, s - 1]] l[s - 1])/(a[[s, s - 1]] d[s - 1] + a[[s, s]]),
    {s, 2, n - 2}]];
```

Решим, теперь, систему (5) методом прогонки при $k = 0$, j меняется от 1 до $n-1$. Сначала вычисляем свободные члены системы (5), для каждой пары $k = 0, j$, затем определяем прогоночные коэффициенты и выполняем обратную прогонку:

In[8]:=
```
k = 0; Do[
 bkh[j, k];
 prog[j, k];
 u[n - 1, j, k + 1/2] = (bkh[j, k][[n - 1]] - a[[n - 1, n - 2]] l[n - 2])/
   (a[[n - 1, n - 2]] d[n - 2] + a[[n - 1, n - 1]]);
 Do[u[i, j, k + 1/2] = d[i] u[i + 1, j, k + 1/2] + l[i],
```

```
{i, n  − 2, 1, −1}],
{j, 1, n − 1}]
```

В результате находим все значения температур в первом промежуточном временном слое:

In[9]:=
```
per = Table[u_{i,j,k+1/2}, {j, 0, n}, {i, 0, n}];
```

Для определения значения температур в первом временном слое сделаем замену производных в дифференциальном уравнении (3) по следующим формулам:

$$\frac{\partial u}{\partial t} \approx \frac{u_{i,j,k+1} - u_{i,j,k+\frac{1}{2}}}{t},$$

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i-1,j,k+\frac{1}{2}} - 2u_{i,j,k+\frac{1}{2}} + u_{i+1,j,k+\frac{1}{2}}}{h^2},$$

$$\frac{\partial^2 u}{\partial y^2} \approx \frac{u_{i,j-1,k+1} - 2u_{i,j,k+1} + u_{i,j+1,k+1}}{h^2}.$$

После преобразований получим следующие разностные уравнения:

$$u_{i,j-1,k+1} - (2 + \frac{2}{\gamma})u_{i,j,k+1} + u_{i,j-1,k+1} =$$

$$(2 - \frac{2}{\gamma})u_{i,j,k+\frac{1}{2}} - u_{i-1,j,k+\frac{1}{2}} - u_{i+1,j,k+\frac{1}{2}} + \frac{t}{\gamma} f[x_i, y_j, t_{k+\frac{1}{2}}]. \quad (6)$$

При фиксированном k и фиксированном i получим систему из $n-1$ разностного уравнения с $n-1$ неизвестной (две неизвестные определены из начальных условий). Это так же трехдиагональная система с той же матрицей системы a. Решаем ее также методом прогонки.

Вводим команду, вычисляющую вектор свободных членов системы (6):

In[10]:=
```
bk[i_, k_] := Block[{b = Table[0, {n − 1}]},
b[[1]] = −u_{i,0,k+1} − u_{i−1,1,k+1/2} − u_{i+1,1,k+1/2} +
(2 − 2/γ)u_{i,1,k+1/2} − t/γ f[x_i, y_1, t_{k+1/2}];
b[[n − 1]] = −u_{i,n,k+1} − u_{i−1,n−1,k+1/2} − u_{i+1,n−1,k+1/2} +
(2 − 2/γ)u_{i,n−1,k+1/2} − t/γ f[x_i, y_{n−1}, t_{k+1/2}];
Do[b[[j]] = −u_{i−1,j,k+1/2} − u_{i+1,j,k+1/2} + (2 − 2/γ)u_{i,j,k+1/2} −
t/γ f[x_i, y_j, t_{k+1/2}],
{j, 2, n − 2}];
b]
```

Решаем системы при $k = 0$:

In[11]:=
```
k = 0; Do[
bk[i, k];
prog[i, k];
u_{i,n−1,k+1} = (bk[i, k][[n − 1]] − a[[n − 1, n − 2]]l_{n−2}) / (a[[n − 1, n − 2]]d_{n−2} + a[[n − 1, n − 1]]);
Do[u_{i,j,k+1} = d_j u_{i,j+1,k+1} + l_j,
{j, n − 2, 1, −1}],
{i, 1, n − 1}]
```

И находим значения температур на первом временном слое:

In[12]:=
```
per = Table[u_{i,j,k+1}, {j, 1, n − 1}, {i, 1, n − 1}]
```

Теперь команды 8 и 11 надо повторить при $k = 1, 2, ..., m$. Сделаем это с помощью команды цикла:

```
In[13]:= Do[Do[
          bkh[j, k];
          prog[j, k];
          u_{n-1,j,k+1/2} = (bkh[j, k][[n - 1]] - a[[n - 1, n - 2]] l_{n-2}) /
            (a[[n - 1, n - 2]] d_n - 2 + a[[n - 1, n - 1]]);
          Do[u_{i,j,k+1/2} = d_i u_{i+1,j,k+1/2} + l_i,
          {i, n - 2, 1, -1}],
          {j, 1, n - 1}];
          Do[
          bk[i, k];
          prog[i,  k];
          u_{i,n-1,k+1} = (bk[i, k][[n - 1]] - a[[n - 1, n - 2]] l_{n-2}) /
            (a[[n - 1, n - 2]] d_{n-2} + a[[n - 1, n - 1]]);
          Do[u_{i,j,k+1} = d_j u_{i,j+1,k+1} + l_j,
          {j, n - 2, 1, -1}],
          {i, 1, n - 1}],
          {k, 0, m - 1}]
```

В результате получим значения всех нужных температур:

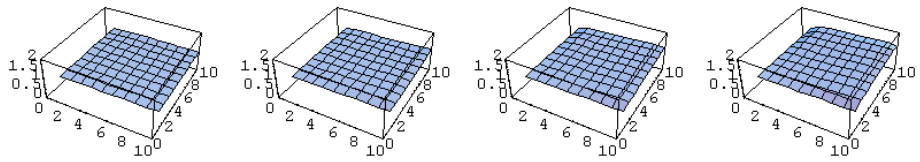
```
In[14]:= per = Flatten[Table[u_{i,j,k}, {k, 0, m}, {j, 0, n}, {i, 0, n}]];
```

Для наглядности введем список значений температур на k-ом слое:

```
In[15]:= g[k_] := Table[u_{i,j,k}, {j, 0, 10}, {i, 0, 10}]
```

и построим приближенно графики функций $u = u(x, y, t)$ для значений времени $t = 0, 1, ..., m$, идущего с шагом 3:

```
In[16]:= p = Table[ListPlot3D[g[k], PlotRange →
                {{0, n}, {0, n}, {0, 2}}], {k, 0, m, 3}];
```



§27 Численные методы оптимизации

Пусть функция $f(x)$ определена на множестве D. Определение точки $x^* \in D$ такой, что

$$f(x^*) = \min_{x \in D} f(x)$$

является основной задачей оптимизации. Если функция $f(x)$ есть функция одной переменной, то такая задача есть задача одномерной оптимизации, если $f(x)$ функция многих переменных - задача многомерной оптимизации.

Метод решения задачи оптимизации есть метод k-порядка, если при решении задачи оптимизации применяется производные функции $f(x)$ до k-порядка включительно, $k \geqslant 0$.

Задачу оптимизации будем решать только для унимодальных функций. Дадим определение. Непрерывная функция $f(x)$, определенная на открытом множестве D, называется унимодальной, если в D данная функция имеет единственную точку локального минимума и нет точек локального максимума. Единственная точка локального минимума в D будет, конечно, и точкой минимума унимодальной функции $f(x)$ в D.

n.1 Метод дихотомии

Методом дихотомии ищется минимум унимодальной функции. Данный метод есть метод нулевого порядка, в нем не используются производные функции.

Пусть функция $f(x)$ унимодальна на интервале (a, b). Сузим интервал так, чтобы новый интервал также содержал точку минимума. Повторим процедуру до тех пор, пока длина нового интервала не станет меньше заданной точности. Любая точка такого интервала и будет решением. Опишем процедуру сужения интервала.

Разобьем интервал на четыре равные части точками:

$$x_0 = \frac{a+b}{2}, \qquad x_1 = \frac{a+x_0}{2}, \qquad x_2 = \frac{x_0+b}{2}$$

и рассмотрим варианты

а) если $f(x_1) \leqslant f(x_0) < f(x_2)$, то точка минимума принадлежит отрезку $x^* \in [a, x_0]$. Сужаем интервал, положим $b = x_0$. Тогда точка x_1 будет серединой нового интервала, поэтому для выполнения следующей итерации можно положить $x_0 = x_1$, построить две точки x_1 и x_2 для нового интервала и найти значения функции в этих точках. Значение функции в точке x_0 можно взять из предыдущей итерации.

б) если $f(x_1) > f(x_0) \geqslant f(x_2)$, то точка минимума принадлежит отрезку $x^* \in [x_0, b]$. Сужаем интервал, положим $a = x_0$. Тогда точка x_2 будет серединой нового интервала, поэтому можно положить $x_0 = x_2$, построить две точки x_1 и x_2 для нового интервала и найти значения функции в этих точках. Значение функции в точке x_0 можно взять из предыдущей итерации.

в) если $f(x_1) > f(x_0)$ и $f(x_0) < f(x_2)$, то точка минимума принадлежит отрезку $x^* \in [x_1, x_2]$. Сужаем интервал, положим $a = x_1$, $b = x_2$. Здесь для следующей итерации нужно вычислять значение функции в двух новых точках - x_1, x_2.

Остальные случаи противоречат унимодальности функции. Так, например случай $f(x_1) = f(x_0) = f(x_2)$ означает, что непрерывная функция $f(x)$ на отрезках $[x_1, x_0]$ и $[x_0, x_2]$ обладает по крайней мере двумя точками, в которых функция $f(x)$ достигает локального минимума.

Теперь проверяем условие остановки $|a - b| < \varepsilon$. Если условие не выполняется, повторяем процесс уменьшения отрезка, при этом, дополнительно нужно построить только две точки и найти значения функции в этих точках.

Найдем минимум унимодальной функции

```
In[1]:= f[x_] = x^2;
```

на интервале

In[2]:= $\mathbf{a = -3;}$
$\mathbf{b = 2;}$
In[3]:= $\mathbf{x_0 = \frac{a+b}{2}//N;}$
$\mathbf{x_1 = \frac{a+x_0}{2}//N;}$
$\mathbf{x_2 = \frac{x_0+b}{2}//N;}$
$\mathbf{m = f[x_1]; n = f[x_0]; p = f[x_2];}$
$\mathbf{i = 1;}$

Составим цикл, применяя команду While. Цикл остановится, когда длина отрезка $[a, b]$ станет меньше 0.001. При этом, любая точка отрезка $[a, b]$ является решением с абсолютной погрешностью 0.001.

```
While[Abs[b − a] > 0.001,
    i + +;
    If[m > n ⩾ p, a = x0; x0 = x2; n = p];
    If[m ⩽ n < p, b = x0; x0 = x1; n = m];
    If[m > n < p, a = x1; b = x2];
    x1 = (a + x0)/2; x2 = (x0 + b)/2;
    m = f[x1]; p = f[x2]
];
Print["На", i, " − ом шаге получен минимум в точке x0 = ", x0]
Print["a = ", a, "x1 = ", x1, "x0 = ", x0, "x2 = ", x2, "b = ", b]
```

На 8-ом шаге получен минимум в точке $x_0 = -0.00012207$

$a = -0.000427246 \quad x_1 = -0.000427246 \quad x_0 = -0.00012207 \quad x_2 = 0.000183105 \quad b = 0.000183105$

Отметим, что в программе значение функции на каждом шаге цикла вычисляется только в двух точках.

n.2 Метод золотого сечения

Метод золотого сечения позволяет найти минимум унимодальной функции и является методом нулевого порядка. Если такой минимум есть и ноль функции (например, для функции x^2), то метод позволяет найти и нули функции.

Пусть унимодальная функция $f(x)$ определена на интервале (a, b). Возьмем точки $x_1, x_2 \in [a, b]$, $x_1 < x_2$ и найдем $f(x_1)$ и $f(x_2)$. Если $f(x_1) < f(x_2)$, то минимум находится на отрезке $[a, x_2]$, в противном случае минимум принадлежит отрезку $[x_1, b]$. В перовом случае положим $b = x_2$, во-втором случае $a = x_1$ и повторим процесс для нового отрезка $[a, b]$. Уменьшаем так отрезок до тех пор, пока его длина не станет меньше заданного $\varepsilon : |b-a| < \varepsilon$. Тогда любая точка отрезка есть минимум функции с абсолютной погрешностью равной ε. При таком подходе, на каждом шаге приходится вычислять значения функции в двух точка. Поэтому такой метод не отличается от метода дихотомии.

Оказывается, можно вычислять значение функции только в одной точке на каждом шаге. Для этого надо специальным образом выбирать точки x_1, x_2.

В методе золотого сечения вычисляется значение функции только в одной точке, а второе значение берется из предыдущей итерации. Дадим определение золотого сечения.

Точка C делит отрезок $[a,b]$ по правилу золотого сечения на отрезки v и u, $v>u$, если

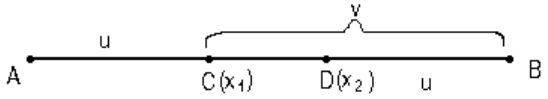
$$u+v=b-a, \quad \frac{b-a}{v}=\frac{v}{u}.$$

Если обозначим $t=\frac{v}{u}>1$, то, из соотношения

$$\frac{u+v}{v}=\frac{v}{u}, \tag{1}$$

получаем $1+\frac{1}{t}=t$ или $t^2-t-1=0$, откуда

$$t=\frac{1+\sqrt{5}}{2}\approx 1.62$$



Пусть точки A, B - концы отрезка $[a,b]$. Пусть точка C имеет координату x_1 и делит отрезок $[a,b]$ в отношении золотого сечения t : $AC=u$, $CB=v$ и $t=\frac{v}{u}$. Отложим на отрезке AB отрезок BD равный u и пусть точка D имеет координату x_2. Тогда $AD=CB=v$. Оказывается, что точка C делит отрезок AD в отношении золотого сечения t. Для доказательства проверим равенства

$$\frac{AD}{AC}=\frac{AC}{CD}$$

или

$$\frac{v}{u}=\frac{u}{v-u}.$$

Но, это равенство равносильно (1). Поэтому, если точка C делит отрезок AD в отношении золотого сечения t, то и точка D делит отрезок CB в отношении t.

Отметим, что из (1) $t=\frac{v}{u}=\frac{u+v}{v}=\frac{b-a}{v}$. Откуда $v=\frac{b-a}{t}$, следовательно координаты точек C и D можно получить так:

$$x_1=b-v=b-\frac{b-a}{t}, \quad \text{а} \quad x_2=a+\frac{b-a}{t}. \tag{2}$$

Опишем теперь первый шаг метода золотого сечения. Возьмем точки с координатами $x_1,x_2\in[a,b]$, найденными по формуле (2). Эти точки делят отрезок $[a,b]$ в отношении золотого сечения. Сравниваем значения $m=f(x_1)$ и $n=f(x_2)$.

Если $m>n$, то то минимум функции находится на отрезке $[x_1,b]$. Уменьшаем отрезок, содержащий минимум функции - положим $a=x_1$. На новом отрезке $[a,b]$ снова выбираем две точки, делящие новый отрезок в отношении золотого сечения. Но, одна такая точка уже есть, это точка x_2, известно и значение функции в этой точке. Точка D (см. рис.) ближе к точке C, чем к точке B, так как $CD=v-u<u=DB$ или $t=v/u<2$. Поэтому полагаем $x_1=x_2$, $m=n$ и находим координаты второй точки $x_2=a+(b-a)/t$, делящую новый отрезок $[a,b]$ в отношении золотого сечения. Теперь вычисляем одно значение функции $n=f(x_2)$.

Если $m \leqslant n$, то минимум функции находится на отрезке $[a, x_2]$. Сужаем отрезок $[a, b]$, положим $b = x_2$. На новом отрезке $[a, b]$ выбираем точки, делящие отрезок в отношении золотого сечения. Одна точка уже есть, переобозначим ее, положив $x_2 = x_1$ (так как точка x_1 ближе к D чем к A). Значение функции в этой точке известно, переобозначим его $n = m$ и строим новую точку $x_1 = b-(b-a)/t$, делящую новый отрезок $[a, b]$ в отношении золотого сечения. Вычисляем одно значение функции $m = f(x_1)$.

Если для нового отрезка $|a - b| < \varepsilon$, то x_1 - искомый минимум, если - нет, то повторяем процесс для отрезка $[a, b]$.

Отметим, что длины так построенных отрезков неограниченно уменьшаются. Действительно, пусть v_i, $i = 0, 1, ..., k$, длина отрезка, содержащего точку минимума, такая, что $v_i/v_{i-1} = t$, $v_0 = b - a$. Перемножая все такие равенства, получим $v_0/v_k = t^k$. Отсюда $v_k = v_0/t^k \to 0$ при $k \to \infty$.

Рассмотрим функцию:

In[1]:= $\mathbf{f[x\_] = (x-2)^2 + 0.2;}$

Введем золотое сечение:

In[2]:= $\mathbf{t = (1+\sqrt{5})/2//N;}$

Концы отрезка:

In[3]:= $\mathbf{a = 0;}$
$\mathbf{b = 3;}$

Вычисляем координаты точек C и D :

In[4]:= $\mathbf{x_1 = b-(b-a)/t;}$
$\mathbf{x_2 = a+(b-a)/t;}$

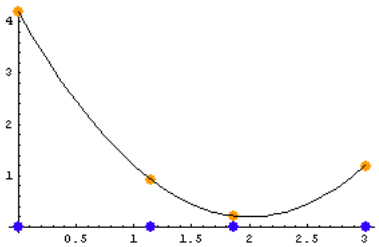
и находим значения функции в этих точках

In[5]:= $\mathbf{m = f[x_1];}$
$\mathbf{n = f[x_2];}$

Построим все точки

In[6]:=
```
gr0 = ListPlot[{{a, 0}, {x1, 0}, {x2, 0}, {b, 0}},
PlotStyle → {PointSize[.03], Hue[.1]}, ];
gr1 = ListPlot[{{a, f[a]}, {x1, m}, {x2, n}, {b, f[b]}},
PlotStyle → {PointSize[.03], Hue[.7]}];
gr2 = Plot[f[x], {x, a, b}];
Show[gr0, gr1, gr2, PlotRange → All];
Print["a = ", a, "x1 = ", x1, "x2 = ", x2, "b = ", b];
```

Out[6]=



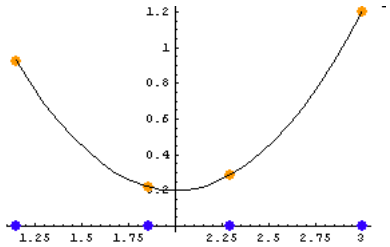
$a = 0 \quad x_1 = 1.1459 \quad x_2 = 1.8541 \quad b = 3$

Видим, что $m = f(x_1) > n = f(x_2)$, следовательно минимум находится на отрезке $[x_1, b]$. Положим $a = x_1$, $x_1 = x_2$, $m = n$, строим новую точку $x_2 = a+(b-a)/t = 2.2918$,

делящую новый отрезок $[1.1459, 3]$ в отношении золотого сечения, и вычисляем одно значение функции $n = f(x_2)$. Построим новый отрезок и функцию

```
In[7]:= a = x1; x1 = x2; m = n; x2 = a + (b - a)/t; n = f[x2];
In[8]:= gr0 = ListPlot[{{a, 0}, {x1, 0}, {x2, 0}, {b, 0}},
        PlotStyle → {PointSize[.03], Hue[.1]}];
        gr1 = ListPlot[{{a, f[a]}, {x1, m}, {x2, n}, {b, f[b]}},
        PlotStyle → {PointSize[.03], Hue[.7]}];
        gr2 = Plot[f[x], {x, a, b}];
        Show[gr0, gr1, gr2, PlotRange → All];
        Print["a = ", a, "x1 = ", x1, "x2 = ", x2, "b = ", b];
```

Out[8]=



$a = 1.1459 \quad x_1 = 1.8541 \quad x_2 = 2.2918 \quad b = 3$

Продолжаем итерационный процесс до достижения заданной точности.

Воспользуемся командой While для организации цикла. Счетчик k показывает число проходов цикла, погрешность $\varepsilon = 0.001$.

Перед выполнением следующей команды нужно ввести команды 1-5.

```
In[6]:= k = 0; While[Abs[b - a] > 0.001,
        k++;
        If[m > n,
        a = x1; x1 = x2; x2 = a + (b - a)/t; m = n; n = f[x2],
        b = x2; x2 = x1; x1 = b - (b - a)/t; n = m; m = f[x1]]];
        Print["Минимум в точке x1 = ", x1,
        "получен на", k, "шаге."]
```

Out[6]= Минимум в точке $x_1 = 1.99993$ получен на 17 шаге.

n.3 Методы оптимизации первого и второго порядков

Пусть унимодальная функция $f(x)$ дважды непрерывно дифференцируема на интервале (a, b). Для определения такой единственной точки минимума достаточно решить уравнение $f'(x) = 0$. Решение такого уравнения можно найти, применяя, например, метод деления отрезка пополам (см. §13). Так как при этом используется только первая производная функции $f(x)$, то такой метод оптимизации относится к методам первого порядка.

Для решения уравнения $f'(x) = 0$ можно применить метод Ньютона (см. §13). При этом используются первые две производные функции $f(x)$, поэтому такой метод оптимизации есть метод второго порядка.

n.4 Многомерная оптимизация

Для нахождения минимума унимодальной, один раз дифференцируемой функции нескольких переменных $f(x_1, ..., x_n)$, можно применить метод градиентного спуска (см. §16).

Если функция дважды непрерывно дифференцируема, то точку минимума унимодальной функции можно найти, решая систему уравнений, составленную из частных производных такой функции:

$$\frac{\partial f(x_1, ..., x_n)}{\partial x_i} = 0, \qquad i = 1, 2, ..., n.$$

Нелинейную систему можно решить, например, методом Ньютона (см. §15).

§28 Темы лабораторных работ

Лабораторная работа 1. Погрешности вычислительных операций.

Здесь и далее G - номер группы, N - номер студента в списке группы.

1. Определить количество верных знаков в числе $a = 0.1GN$, если относительная погрешность числа - $a = G\%$. Определить Δ_a - абсолютную погрешность числа a и округлить его до верных знаков.

2. Даны три приближенные числа: $a = 0.01GN$, $b = 0.1N$, $c = 0.1(G + 10)$. Найти точные значения следующих величин: $s = a+b+c$; $d = \frac{ac-b}{c}$, $y = e^a\sqrt{b^2+c^2}$. Оценить погрешности результатов, считая, что все знаки в a, b, c - верные.

3. Вычислить и оценить погрешность произведения bc если $\Delta_b = 0.001N$, $\Delta_c = 0.001G$.

Лабораторная работа 2. Сгенерировать[1] квадратную матрицу A с преобладанием диагональных элементов и столбец свободных членов b порядков $n = 7 + N$, N-номер студента.

Решить систему $A.x = b$:

1. Методом Гаусса.

2. Методом Гаусса с выбором главного элемента.

3. Используя вычислительную схему какого-либо из методов исключения вычислить - определитель матрицы системы .

4. Найти обратную матрицу к матрице A методом Жордана.

Лабораторная работа 3. Методы Якоби, Зейделя и прогонки. Сгенерировать трехдиагональную квадратную матрицу A с преобладанием диагональных элементов и столбец свободных членов b порядков $n = 7 + N$, N-номер студента.

1. Решить систему $A.x = b$ с помощью:

а) метода Якоби;

б) метода Зейделя.

в) методом прогонки.

Найти вектор невязки $r = b - A.x^*$ для решения x^* и вычислить норму этого вектора.

Лабораторная работа 4. LU-разложение матрицы.

Сгенерировать квадратную матрицу A с преобладанием диагональных элементов порядка $n = 7 + N$, N-номер студента.

1. Построить LU-разложение матрицы A.

2. Сгенерировать столбец свободных членов b и решить систему $A.x = b$ с использованием LU-разложение матрицы A. Найти невязку решения.

3. Вычислить определитель матрицы A и найти для нее обратную матрицу, используя LU-разложение матрицы A.

Лабораторная работа 5. Метод квадратных корней.

Сгенерировать симметричную квадратную матрицу A порядка $n = 7 + N$, N-номер студента, и столбец свободных членов b и решить систему $A.x = b$ методом квадратных корней. Найти невязку решения.

Лабораторная работа 6. Методы вращения и отражения.

1. Сгенерировать квадратную матрицу A порядка $n = 7+N$, N-номер студента, и столбец свободных членов b и решить систему $A.x = b$ методом вращений. Найти

[1] См., например, матрицу m на стр. 37.

невязку решения.

2. Решить данную систему методом отражений. Найти невязку решения.

Лабораторная работа 7. Обращение матриц.

1. Сгенерировать квадратную матрицу A порядка $n = 7+N$, N-номер студента и найти обратную матрицу методами окаймления и пополнения.

Лабораторная работа 8. Градиентные методы.

1. Сгенерировать квадратную матрицу A с преобладанием диагональных элементов и положительных диагональных элементов порядка $n = 7 + N$, N-номер студента, и столбец свободных членов b и решить систему $A.x = b$ методом наискорейшего спуска. Найти невязку решения.

2. Решить данную систему методом сопряженных градиентов.

Лабораторная работа 9. Системы с прямоугольными матрицами.

1. Сгенерировать матрицу A порядка $m \times n$, $m = 7 + N > n = 7$, N-номер студента, и столбец свободных членов b и найти обобщенное решение системы A.x=b

2. Сгенерировать матрицу A порядка $m \times n$, $n = 7 + N > m = 7$, N-номер студента, и столбец свободных членов b и найти нормальное решение системы $A.x = b$.

3. Найти псевдообратную матрицу для матрицы из п. 1.

Лабораторная работа 10. Собственные значения симметричной матрицы.

1. Применяя метод Якоби, найти собственные векторы и значения симметричной матрицы системы из лабораторной работы 3.

2. Методом Гивенса привести матрицу из п.1 к трехдиагональному виду и найти собственные значения матрицы методом бисекций.

Лабораторная работа 11. Частичная проблема собственных значений. Степенной метод.

Сгенерировать квадратную матрицу A с преобладанием диагональных элементов порядка $n = 7+N$, N-номер студента и найти максимальное по модулю собственное значение матрицы A

Решить частичную проблему собственных значений - найти N-ое собственное значение методом бисекций из 10 лабораторной работы

Лабораторная работа 12. QR-разложение. Решение полной проблемы собственных значений произвольной матрицы.

Сгенерировать квадратную матрицу A порядка $n = 7 + N$, N-номер студента, и найти все собственные значения матрицы A, применяя QR-разложение.

Лабораторная работа 13. Решение нелинейных уравнений.

Дано уравнение: $2^x + Nx - 10 = 0$, N - номер студента.

1. Найти промежутки изоляции корней уравнения графически.

2. Вычислить значение одного из решений уравнения с точностью $\varepsilon = 0.001$:

а) методом половинного деления;

б) методом касательных:

в) методом простой итерации.

Лабораторная работа 14. Решение систем нелинейных уравнений

Дана система нелинейных уравнений:

$$\begin{cases} \sin\left(\frac{x}{3} - 1\right) + y = 0.1N, \\ Gx + \cos\frac{y}{3} = G, \end{cases}$$

где G- номер группы, N - номер студента в списке группы.

Найти решение системы с точностью до 0.001 используя:

а) метод итераций;

б) метод Ньютона;

и) упрощенный метод Ньютона.

Лабораторная работа 15. Метод градиентного спуска.

Найти решение системы уравнений

$$\begin{cases} \frac{1}{G}x + y^2 - 2 = 0, \\ y - x^2 + 3 = 0. \end{cases}$$

методом градиентного спуска.

Лабораторная работа 16. Интерполяция и аппроксимация таблично заданных функций.

1. Построить таблицу значений функции $f(x) = E^{\frac{x}{G}} \cos(x + N)$, где G - номер группы, N - номер студента в списке группы, на промежутке $[-G, G]$ с шагом $h = 0.4G$.

2. Построенную табличную функцию доопределить в точке $x = 0.3G$, используя интерполяционный полином Лагранжа 5-го порядка для равноотстоящих узлов и интерполяционный полином Лагранжа 5-го порядка для неравноотстоящих узлов, найденных с помощью корней многочлена Чебышева. Сравнить полученные значения со значением функции $f(0.3G)$.

4. Построенную табличную функцию доопределить в точке $x = 0.3G$, используя интерполяционные многочлены Ньютона для интерполирования вперед и назад. Сравнить результаты вычислений.

3. Для той же табличной функции осуществить аппроксимацию методом наименьших квадратов полиномом 4-го порядка. Используя полученную аппроксимацию, доопределить табличную функцию в точке $x = 0.3G$.

4. Построить дважды непрерывно дифференцируемый кубический сплайн по данной табличной функции, то есть построить сплайн дефекта 1.

Лабораторная работа 17. Численное интегрирование.

С точностью $\varepsilon = 0.001$ вычислить приближенное значение интеграла

$$\int_0^b \sin\left(\frac{x^2}{G} + 0.1N\right)dx,$$

где G - номер группы, N - номер студента в списке группы, $b = \frac{\pi}{2}G$:

а) по формуле прямоугольников;

б) по формуле трапеций;

с) по формуле Симпсона;

д) по формулам Ньютона-Котеса;

е) применяя квадратурные формулы Гаусса-Кристоффеля.

Для вычисления интеграла каждым из методов найти шаг таблицы и построить таблицу значений подынтегральной функции с этим шагом на промежутке $[0, b]$.

Лабораторная работа 18. Численные методы решения задачи Коши

Дана задача Коши: $y' = Nx^2 + y^2$, $y(0) = 0.N$, N - номер студента в списке группы.

1. Решить задачу при помощи метода Эйлера на промежутке $[0; 1]$ с шагом $h = 0.1$.

2. Решить задачу при помощи метода Рунге - Кутта на промежутке $[0;1]$ с шагом $h = 0.1$.

Скорректировать интервал, если потребуется.

Лабораторная работа 19. Системы дифференциальных уравнений.

Решить следующую задачу Коши, преобразовав уравнение в систему дифференциальных уравнений первого порядка.

$$y''''' + \frac{1}{x+G}y'''' - \frac{1}{(x+G)^2}y''' + x^2y''' - (x+y)y'' - y' + y + 6x^2 = 0.1N,$$

$y(0) = y'(0) = y''(0) = y'''(0) = y''''(0) = 1$, где G - номер группы, N - номер студента в списке группы.

Решить задачу на промежутке $[0;1]$ с шагом $h = 0.1$.

Лабораторная работа 20. Разностным методом решить задачу Коши:

$$y'' + \frac{1}{x+G}y' - \frac{1}{(x+G)^2}y = 0.1N,$$

$y(0) = y'(0) = 1$, где G - номер группы, N - номер студента в списке группы.

Решить задачу при помощи метода конечных разностей на промежутке $[0;1]$ с шагом $h = 0.1$. Воспользоваться встроенной командой для определения решения данной дифференциальной задачи и скорректировать, если потребуется, интервал, шаг и начальные условия для поставленной задачи.

Лабораторная работа 21. Метод переменных направлений.

Применяя метод переменных направлений найти численное решение двумерной задачи теплопроводности:

$$\frac{\partial u}{\partial t} = a^2\Big(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\Big) + f(x,y,t);$$

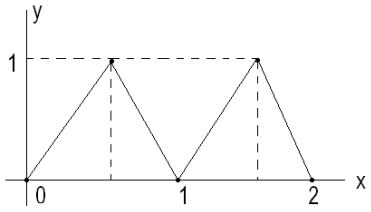
$\varphi(x,y) = 0$, $\psi(x,y,t) = \ln(1+xyz)$, $f(x,y,t) = (1+t^2)\sin(\pi y)$, $a = 1$, $h = 0.25$, $T = 1$, $\tau = T/N$, N - номер студента. Область G есть параллелограмм с вершинами в точках $(0,0)$, $(1,1)$, $(3,1)$, $(2,0)$.

Лабораторная работа 22. Уравнение колебания струны с закрепленными концами.

Применяя разностный метод, найти численное решение уравнения колебания струны длины $l = 2$:

$$\frac{\partial^2 u}{\partial t^2} = a^2\frac{\partial^2 u}{\partial x^2}$$

с $a = 2$, начальным профилем



начальной скоростью:

$$\frac{\partial u}{\partial t}(x,0) = 0$$

и граничными условиями:

$$u(0,t)=0, \quad u(l,t)=0.$$

Лабораторная работа 23. Численные методы оптимизации.

Найти минимум функции $y=G^2(x-N)^4+N$ методами дихотомии и золотого сечения.

§29 Приложение. Команды пакета Mathematica 6

Все внутренние команды и функции Математики начинаются с прописной буквы, аргументы всегда заключены в квадратные скобки.

Обозначение основных математических операций в Математике традиционно: +, −, ∗ (умножение можно заменять пробелом), / (деление), ∧ (возведение в степень). Знак = означает присваивание, например, $a=3$. Результатом логических операции есть $True$ или $False$: == (равенство), ! = (неравенство), > (больше), < (меньше), >= (больше или равно), <= (меньше или равно), && логическое "и", || логическое "или". Каждая операция имеет другое обозначение. Например, математическая операция "+" - Plus[x,y]; x,y - числа, логическое "и" - And[x,y], логическое "или" - Or[x,y]; x,y - логические высказывания.

<< **RealTime3d`** - включает режим вращения трехмерного графического образа мышкой в программе Mathematica 5.

<< **Default3d`** -отмена режима вращения трехмерного графического образа мышкой в программе Mathematica 5. Отметим, что здесь апостроф берется с клавиши с тильдой " ∼ ".

Animate[expr,{i,1,n}] - генерирует анимацию выражения expr при непрерывном изменении i от 1 до n.

ArcSin[**x**], **ArcCos**[**x**], **ArcTan**[**x**] - обратные тригонометрические функции.

Append[list,element] добавляет *element* в конец списка *list* (список *list* при этом не меняется).

Apply[f, expr] или **f@@expr** замещает заголовок выражения *expr* на f.

AspectRatio−> $\frac{\mathbf{m}}{\mathbf{n}}$ - опция для команд Plot и Show, определяющая пропорцию размеров двумерного графика - отношение высоты к ширине $\frac{m}{n}$. По умолчанию задано золотое сечение 1/GoldenRatio.

AxesOrigin−> $\{\mathbf{x_0},\mathbf{y_0}\}$ - опция для двумерной графики, указывающая команде Plot координаты точки пересечения осей.

Block и **Module**. Формат команд одинаков: **Block[{x,y,z,...},body]** В скобках указаны имена локальных переменных (во втором случае и их начальные значения) и участок программы *body*, где эти переменные являются локальными.

body&, где **body** обычная функция, в которой вместо первой переменной записан знак #1, второй - #2, и так далее. Такие функции называются чистыми (pure) функциями.

Count[list,form] находит число вхождений $form$ в список.

Clear[x,y,z,...] - очищает значения переменных $x, y, z, ...$

Complement[**list, l1,l2,...**] убирает в *list* элементы, принадлежащие $l1$, $l2$,.....

Cross[**a,b**] - векторное произведение трехмерных векторов a и b, заданных своими координатами, как списки.

CharacteristicPolynomial[**m, x**] находит характеристический многочлен матрицы m с переменной x.

Dimensions[**list**] возвращает размерность списка.

DeleteCases[**list, pattern**] удаляет из списка *list* элементы по образцу pattern;

Drop[**list, n**] удаляет из списка *list* первые n элементов.

Drop[**list, -n**] удаляет из списка *list* последние n элементов.

Drop[**list, {n}**] удаляет n-ый элемент списка *list*.

Drop[**list, {m, n}**] удаляет из списка *list* элементы от m до n.

Det[**m**] вычисляет определитель квадратной матрицы m.

DiagonalMatrix[**list**] создает диагональную матрицу с главной диагональю, сформированной из элементов списка *list*, и 0 - остальные элементы матрицы.

Dot[**a, b, c**] находит произведение векторов, матриц и тензоров. Операцию произведения можно задавать также в виде $a.b$.

D[**f**[**x**], **x**] или **f'**[**x**] вычисляет производную функции $f[x]$ по x.

D[**f**[**x**], **{x,n}**] вычисляет n-ую производную функции $f[x]$ по x. Первые производные можно обозначать штрихами: $f'[x]$, $f''[x]$,...

D[**f**[$\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}$], {$\mathbf{x_i}, \mathbf{n}$}] вычисляет частную производную n-го порядка функции $f[x_1, x_2, ..., x_n]$ по переменной x_i, $i = 1, 2, ..., n$.

D[**f**[$\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}$], **x1**, **x2**, ...] вычисляет производную функции $f[x_1, x_2, ..., x_n]$ по переменным $x_1, x_2, ...,$. Переменные могут повторяться и располагаться в любом порядке.

DSolve[**eqns, y**[**x**], **x**], список eqns содержит дифференциальное уравнение относительно функции y[x] и, при необходимости, начальные условия, записанные как уравнения: $y[x_0] == y_0$, $y'[x_0] == y_0, ...$ Если eqns состоит только из дифференциального уравнения, то фигурные скобки, обозначающие список, можно не писать. Решение дифференциального уравнения записывается в виде $\{\{y[x] -> expr\}\}$, означающем, что решением является функция $y[x] = expr$. При этом функция *expr* может быть чистой функцией одной переменной или интерполяционным многочленом.

DSolve[{$\mathbf{eqn_1}, \mathbf{eqn_2}, ..., \mathbf{eqn_n}$}, {$\mathbf{y_1}$[$\mathbf{x_1}, ..., \mathbf{x_m}$], ..., $\mathbf{y_n}$[$\mathbf{x_1}, ..., \mathbf{x_m}$]}, $\{x_1, ..., x_m\}$] решает систему дифференциальных уравнений. В список могут входить и начальные условия.

Do[**expr, {i, a, b, h}**] - выполняет команды списка *expr*, разделенных точкой с запятой, при каждом изменении счетчика цикла i от a до b с шагом h. Если h не писать, то шаг изменения счетчика i равен $+1$.

Exp[**x**] - e^x.

Eigensystem[**N**[**m**]] формирует список $\{values, vectors\}$ собственных значений и собственных векторов данной числовой матрицы m.

Eigenvalues[**N**[**m**]] формирует список собственных значений числовой матрицы m.

Eigenvectors[**N**[**m**]] формирует список собственных векторов числовой матрицы m.

Expand[**expr**] раскрывает произведения и положительные целые степени в *expr*.

Flatten[**list**] выравнивает (превращает в одномерный) список данный вложенный список *list*.

FreeQ[list,form] проверяет, свободен ли список от $form$: если да - возвращает $True$, иначе возвращает $False$.

FullSimplify, применяемая так же как и $Simplify$, подвергает выражение более широкому классу преобразований.

GraphicsArray[{p,q,...}] - размещает графики $p, q, ...$ горизонтально, а если список графиков есть двумерный список, то графики располагаются как элементы матрицы.

Integrate[f[x],x] находит первообразную функции $f[x]$.

Integrate[f[x], {x, x$_{min}$, x$_{max}$}] находит значение определенного интеграла функции f[x] при x, изменяющемся от x_{min} до x_{max}.

IntederPart[x] целая часть числа x.

IdentityMatrix[n] создает единичную матрицу с размером $n \times n$ (на главной диагонали стоят 1, остальные элеиенты - 0).

Inverse[m] находит обратную матрицу для квадратной невырожденной матрицы m.

<< Graphics`ImplicitPlot` - команда загружает программу построения множеств заданных неявно в пакте Mathematica 5. Необходимо загрузить программу для построения одного множества нулей функции:

ImplicitPlot[f[x, y] == 0, {x, x$_{min}$, x$_{max}$}, {y, y$_{min}$, y$_{max}$},
options–> value]

или нескольких множеств нулей функций:

ImplicitPlot[{f$_1$[x, y] == 0, f$_2$[x, y] == 0, ...},
{x, x$_{min}$, x$_{max}$}, options–> value]

Join[list1, list2, ...] объединяет списки в единую цепочку (выполняет конкатенацию).

k++, k-- - краткая запись соответственно $k = k + 1$ и $k = k - 1$.

Length[list] возвращает число элементов списка list.

Last[list] возвращает последний элемент списка $list$.

Log[x], **Log[x, a]** - натуральный логарифм, логарифм x по основанию a.

ListPlot[{{x1, y1}, {x2, y2}, ...}] строит в координатной плоскости точки с координатами (x_1, y_1), (x_2, y_2), ...

ListPlot[{y$_1$, y$_2$, ...}] строит точки с координатами $(1, y_1)$, $(2, y_2)$, ...

ListPlot3D[m] рисует поверхность, заданную прямоугольной матрицей m, элементы которой есть высоты узловых точек поверхности.

LinearSolve[A, B] возвращает вектор X - решение матричного уравнения A.X=B.

MemberQ[list,form] проверяет, есть ли $form$ в списке, и возвращает $True$, если это так и $False$ в противном случае.

MatrixQ[m], **MatrixQ[m, NumberQ]** проверяют является ли m - матрицей, числовой матрицей.

Max[list], **Min[list]** находят максимальный и минимальный элемент списка $list$.

Minors[m, k] создает вложенный список всех миноров порядка $k \times k$ матрицы m. Внутренние списки состоят из всех миноров k-го порядка матрицы m, элементы которых принадлежат одним и тем же строкам матрицы m.

m[[i$_1$;; i$_2$, j$_1$;; j$_2$]] возвращает матрицу из элементов, стоящих на пересечении $i_1, ..., i_2$ строк и $j_1, ..., j_2$ столбцов. В Mathematica 5 эта команда не работает (см. примечание на стр. 15).

m[[**i**;;,;;**j**]] возвращает матрицу из элементов, стоящих на пересечении $i, ..., n$ строк и $1, ..., j$ столбцов. Аналогично работают команды **m**[[;;**i**,**j**;;]], **m**[[**i**;;,**j**;;]] и так далее.

m[[**i**;;**i**,**j**$_1$;;**j**$_2$]] возвращает матрицу-строку из элементов $j_1, ..., j_2$ i-ой строки, то есть матрицу вида $\{\{a_{ij_1}, ..., a_{ij_2}\}\}$.

m[[**i**,**j**$_1$;;**j**$_2$]] возвращает вектор из элементов $j_1, ..., j_2$ i-ой строки, то есть вектор вида $\{a_{ij_1}, ..., a_{ij_2}\}$.

m[[**i**$_1$;;**i**$_2$,**j**;;**j**]] возвращает матрицу-столбец из элементов $i_1, ..., i_2$ j-ого столбца, то есть матрицу вида $\{\{a_{ij_1}\}, ..., \{a_{ij_2}\}\}$.

m[[**i**$_1$;;**i**$_2$,**j**]] возвращает вектор из элементов $i_1, ..., i_2$ j-ого столбца, то есть вектор вида $\{a_{i_1,j}, ..., a_{i_2,j}\}$.

m[[{**i**$_1$,**i**$_2$,...,**i**$_k$},{**j**$_1$,**j**$_2$,...,**j**$_l$}]] возвращает минор матрицы m состоящий из элементов, стоящих на пересечении $i_1, i_2, ..., i_k$ строк и $j_1, j_2, ..., j_l$ столбцов. **m**[[{**i**$_1$,**i**$_2$,...,**i**$_k$}]] возвращает минор матрицы m состоящий из элементов $i_1, i_2, ..., i_k$ строк.

m[[**All**,{**i**$_1$,**i**$_2$,...,**i**$_k$}]] есть минор матрицы m состоящий из элементов $i_1, i_2, ..., i_k$ столбцов.

Map[f, expr] или **f/@expr** применяет f к каждому элементу на первом уровне в *expr* в следующем смысле:

Map[f, expr, levelspec] применяет f к частям *expr*, указанным с помощью *levelspec*.

N[expr] (или **exp//N**) - дает численное значение арифметического выражения expr или является предикатом, если выражение expr есть логическое высказывание.

N[expr,n] - дает численное значение арифметического выражения expr с n знаками после запятой.

n! - **n**-факториал, **Abs**[**x**] - модуль числа x.

NSolve[**f**[**x**] == **0**,**x**] решает уравнение f(x)=0 относительно переменной x.

NSolve[{**f**$_1$[**x**$_1$,**x**$_2$,...,**x**$_n$] == **0**,**f**$_2$[**x**$_1$,**x**$_2$,...,**x**$_n$] == **0**,...,

f$_n$[**x**$_1$,**x**$_2$,...,**x**$_n$] == **0**},{**x**$_1$,**x**$_2$,...,**x**$_n$}] численно решает систему уравнений.

N[**Integrate**[**f**[**x**],{**x**,**x**$_{min}$,**x**$_{max}$}],**n**] находит численное значение определенного интеграла функции f[x] при x, изменяющемся от x_{min} до x_{max} с точностью n десятичных знаков после запятой.

NDSolve[**eqns**,**y**,{**x**,**x**$_{min}$,**x**$_{max}$}], где список eqns содержит дифференциальное уравнение относительно неизвестной функции y[x], $x \in [x_{min}, x_{max}]$ и начальные условия, записанные в виде $y[x_0] == y_0, ...$ Решение такого дифференциального уравнения получается в виде интерполяционной функции.

Position[list,form] возвращает номер позиции $form$ в списке.

Partition[list,n,m] - делит список ss на списки длины n со сдвигом m. Так, первый список будет состоят из первых n элементов списка ss, второй список будет состоять из n элементов списка ss, начиная с $m+1$-го элемента списка ss и так далее.

Prepend[list,element] добавляет *element* в начало списка *list*.

Plot[**f**[**x**],{**x**,**x**$_{min}$,**x**$_{max}$},**option**−> **value**,**option**−> **value**,...] строит график функции $f[x]$ аргумента x в интервале от x_{min} до x_{max}.

Иногда функцию или список функций необходимо подготовить для построения ее графика или графиков командой **Evaluate**. Например,

Plot[**Evaluate**[**Table**[**x**$^{\mathbf{i}}$,{**i**,**1**,**7**}]],{**x**,−**1**,**1**}] строит графики функций из списка $\{x, x^2, x^3, x^4, x^5, x^6, x^7\}$.

Plot[{$\mathbf{f_1}$[x], $\mathbf{f_2}$[x], ...}, {x, $\mathbf{x_{min}}$, $\mathbf{x_{max}}$}, **option**−> **value**, ...] совмещает на одном чертеже графики нескольких функций $f_i[x]$, определенных на одном интервале $[x_{min}, x_{max}]$.

PlotStyle−> {{**1стиль**}, {**2стиль**}, ...} - опция для стиля графиков функций. Вместо значений стилей может стоять $Dashing[\{n_1, n_2, n_3, ...\}]$ (пунктирная линия, состоящая из чередующихся отрезков графика длиной $n_1(\frac{1}{72})$ дюйма и пробела длиной $n_2(\frac{1}{72})$ дюйма...), $Hue[m_1]$ (цвета линии, соответствующий параметру m_1) или $Thickness[m_2]$ означающее, что линия графика будет иметь толщину, равную $m_2(\frac{1}{72})$ дюйма, $0 \le n_1, n_2, n_3, m_1, m_2 \le 1$.

PlotRange−> {{$\mathbf{x_1}$, $\mathbf{x_2}$}, {$\mathbf{y_1}$, $\mathbf{y_2}$}} - опция, указывающая команде, что нужно выводить только те точки графика, которые принадлежат прямоугольнику $x_1 \le x \le x_2$, $y_1 \le y \le y_2$.

Опция $PlotStyle \to PointSize[d]$ в этих командах рисует точки в виде закрашенных кружков диаметра d, а $PlotJoined \to True$ соединяет построенные точки отрезками (по умолчанию - False). Если эта опция установлена, то можно указать стиль линий установкой опции $PlotStyle \to \{value\}$. Значениями value могут быть одно или несколько значений вида $Dashing[\{n_1\}]$, $Hue[n_2]$ и $Thickness[n_3]$.

Plot3D[{**f**[x, y], **s**}, {**x**, **xmin**, **xmax**}, {**y**, **ymin**, **ymax**}] позволяет построить график функции f[x,y]. Можно закрасить график одним цветом - для этого нужно вместо s написать Hue[n] или RGBColor[r,g,b], где $n, r, g, b \in [0, 1]$ - параметры цвета.

Product[**f[i], {i, imin, imax}**] вычисляет произведение значений $f[i]$ при изменении i от $i = imin$ до $i = imax$ с шагом +1.

QRDecomposition[**N**[**m**]] возвращает QR-разложение для числовой матрицы m. Результат представляет собой список $\{q, r\}$, где q - ортогональная матрица, r- верхняя треугольная матрица.

Random[] - псевдослучайное число между 0 и 1,

Random[**тип числа**, {**a**, **b**}] псевдослучайное число между a и b, тип числа указывает на тип псевдослучайного числа: Integer - целое число, **Real** - действительное, **Complex** - комплексное число.

Reverse[**list**] возвращает список с обратным порядком расположения элементов.

ReplacePart[**list,a,n**] заменяет на a n-ый элемент списка $list$.

ReplacePart[**list,a,{i,j}**] заменяет на a элемент вложенного списка $list$ в позиции (i, j), то есть в i-ом элементе списка $list$ заменяет j-ый элемент.

RankMatrix[**m**] - ранг матрицы m.

ReplaceAll[**expr, rule**] или, что тоже самое, **expr/.rule** заменяет в выражении $expr$ переменные по правилу, указанному в $rule$.

Sqrt[x] - квадратный корень.

Sin[x], **Cos**[x], **Tan**[x] - тригонометрические функции.

Select[**list,patten**] - выбирает из списка $list$ элементы по правилу, указанному в **patten**.

Show[**p,q**] выводит на экран два графика p, q одновременно.

Solve[**f**[x] == **0**, x] решает уравнение вида f(x)=0 относительно переменной x. Команда возвращает строчку $\{\{x{-}{>}\ a1\}, \{x{-}{>}\ a2\}, ...\}$, означающую, что $x = a1$, $x = a2, ...$ есть решения уравнения. Если функция f(x) содержит число с десятичной точкой или в конце команды стоит //N, то численные решения будут числами с десятичной точкой.

Solve[{$\mathbf{f_1}$[$\mathbf{x_1}$, $\mathbf{x_2}$, ..., $\mathbf{x_n}$] == **0**, $\mathbf{f_2}$[$\mathbf{x_1}$, $\mathbf{x_2}$, ..., $\mathbf{x_n}$] == **0**, ..., $\mathbf{f_n}$[$\mathbf{x_1}$, $\mathbf{x_2}$, ..., $\mathbf{x_n}$] == **0**},

$\{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}\}]$
решает систему уравнений

$$\begin{cases} f_1(x_1, x_2, ..., x_n) = 0, \\ f_2(x_1, x_2, ..., x_n) = 0, \\ \dots \\ f_n(x_1, x_2, ..., x_n) = 0 \end{cases}$$

относительно переменных x_1, $x_2, ..., x_n$. Решение записывается в виде $\{\{x->\ a_1\}$, $\{x-> a_2\}, ..., \{x-> a_n\}\}$.

Solve[**A.X == B, X**], решает матричное уравнение - находит вектор-решение X системы уравнений.

Sum[**f[i], {i, imin, imax}**] начинает суммирование при i изменяющемся от $i = imin$ до $i = imax$ с шагом $+1$.

Sum[**f[i,j], {i, imin, imax}, {j, jmin, jmax}, ...**] проводит суммирование f[i,j] по двум индексам.

Simplify[**expr**] или **expr//Simplify** или **Simplify**[**%n**] пытается привести математическое выражение *expr* (или $\%n$) к наиболее простой форме.

Sort[**list**] сортирует список *list* по возрастанию элементов.

Table[**expr, {m}**] создает список, содержащий m экземпляров выражения *expr*.

Table[**expr, {i, m}**] создает список значений expr при i, изменяющемся от 1 до m.

Table[**expr, {i, m, n}**] - i изменяется от m до n.

Table[**expr, {i, m,n, d}**] - i изменяется от m до n с шагом d.

Table[**expr, {i, m1, n1}, {j, m2, n2 }, ...**] возвращает вложенный список. Внешним является список по переменной i. Число индексов $i, j, ...$ есть размерность вложенного списка. Если размерность списка равна 2, то данный список есть матрица, i - номер строки матрицы, j - номер столбца, а (i, j) элемент матрицы есть j элемент i списка.

Если список обозначим через ls, то $ls[[i]]$ есть i-ый элемент списка, в частности, если ls - матрица, то - i строка матрицы. Элемент $ls[[i, j]]$ есть j элемент i - го списка списка ls.

TableForm или **MatrixForm** (**TableForm**[**list**] или **MatrixForm**[**list**]). Эти команды можно записать как опцию в конце команды, результатом работы которой есть список. Например, чтобы получить произведение матриц a.b в виде матрицы, нужно записать: a.b// MatrixForm.

Take[**list, n**] возвращает первые n элементов списка *list*.

Take[**list, {n}**] возвращает n-ый элемент списка *list*.

Take[**list, -n**] возвращает последние n элементов списка *list*.

Take[**list, {m, n}**] возвращает элементы списка с порядковыми номерами от m до n.

Take[**list, {m, n, s}**] возвращает элементы списка с порядковыми номерами от m до n с шагом s.

Transpose[**list**] транспонирует (переставляет строки и столбцы) двумерные списки (матрицы).

ToString[**expr**] выражение *expr* конвертирует в строку.

Union[**list1, list2**] объединяет списки, убирая дублирующие элементы.

Union[**list**] убирая дублирующие элементы в *list*.

While[**test, expr**] выполняет команды списка *expr*, разделенных точкой с запятой, до тех пор пока значением логической функции *test* является $True$.

Предметный указатель

Литература

[1] Березин И.С., Жидков Н.П. Методы вычислений I,II.- М: Физматгиз, 1960 г.

[2] Васильев Ф.П., Лекции по методам решения экстремальных задач.- Изд.МГУ,1974 г.

[3] Вержбицкий В.М. Основы численных методов.- М: 2002 г.

[4] Волков Е.. Численные методы.- М: 1987 г.

[5] Демидович Б.П., Марон И.А. Основы вычислительной математики. - М: Физматгиз, 1963 г.

[6] Дьяконов В. Mathematica 4. - СПб.: Питер, 2001 г.

[7] Иванов А.П., Олемский И.В., Олемский Ю.В. Численные методы часть I.1. - Учебное пособие, электронный вариант, СПб, 2008 г.

[8] Калиткин Н.Н. Численные методы.- М: Наука, 1978 г.

[9] Корн Г., Корн Т. Справочник по высшей математике.- М: Наука, 1974 г.

[10] Кублановская В.Н. Численные методы алгебры.- Ленинград: Уч. пособие, 1976 г.

[11] Мостовской А.П., Мостовская Л.Г. Компьютерная Математика.- Мурманск: МГПИ, 1998 г.

[12] Мостовской А.П. Информационные технологии в математике, - Мурманск: МГПУ, 2005 г.

[13] Мостовская Л.Г., Середа А.-В.И. Практикум по курсу "Вычислительная математика", часть I.- Мурманск: МГТУ, 2001 г.

[14] Wolfram, S., Mathematica: A system for Doing Mathematics by Computer. - Second Edition, Addison-Wesleg, New York, 1991 г.

[15] Gray John W. Mastering Mathematica: Programming methods and applications. - University of Illinois, Urbana, Illinois, 1994 г.

[16] Программа помощи Mathematica 6.

Оглавление