МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное образовательное бюджетное учреждение
высшего профессионального образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

ЧИСЛЕННЫЕ МЕТОДЫ

Рекомендовано в качестве учебного пособия
Редакционно-издательским советом
Томского политехнического университета

Составитель **О.Л. Крицкий**

Издательство
Томского политехнического университета
2013

УДК 519.95(075.8)
ББК 22.193я73
Ч-67

Ч-67 **Численные методы**: учебное пособие / сост. О.Л. Крицкий; Томский политехнический университет. – Томск: Изд-во Томского политехнического университета, 2013. – 69 с.

В авторской редакции

Пособие подготовлено на кафедре высшей математики и математической физики, соответствует программе дисциплины и предназначено для студентов ИДО, обучающихся по направлению 140400 «Электроэнергетика и электротехника».

УДК 519.95(075.8)
ББК 22.193я73

Рецензенты

Кандидат физико-математических наук, доцент
М.Е. Семёнов

Профессор, заведующий кафедрой
высшей математики и математической физики ТПУ
А.Ю. Трифонов

ОГЛАВЛЕНИЕ

1. НАХОЖДЕНИЕ КОРНЕЙ УРАВНЕНИЙ

Введение

Одним из наиболее распространенных методов поиска корней уравнений является ***метод Ньютона*** и его модификации. Пусть требуется решить уравнение $f(x)=0$. Будем считать, что x является решением уравнения. Разложим функцию $f(x)$ в ряд в точке x_0, близкой к точке x, и ограничимся только первыми двумя членами разложения:

$$f(x)=f(x_0)+f'(x_0)(x-x_0)+\ldots$$

Поскольку x– корень уравнения, то $f(x)=0$. Следовательно,

$$x\approx x_0-\frac{f(x_0)}{f'(x_0)}.$$

Таким образом, если нам известно приближенное значение корня уравнения, то полученное уравнение позволяет его уточнить. Понятно, что процесс уточнения можно повторять многократно, до тех пор, пока значение функции не будут отличаться от нуля на величину меньшую, чем заданная точность поиска. Очередное k-е приближение находится по формуле

$$x_{k+1}=x_k-\frac{f(x_k)}{f'(x_k)}.$$

Ограничившись в разложении только первыми двумя членами, мы фактически заменили функцию $f(x)$ на прямую линию, касательную в точке x_0, поэтому метод Ньютона еще называют методом касательных. Далеко не всегда бывает удобно находить аналитическое выражение для производной функции. Однако в этом и нет особой необходимости; поскольку на каждом шаге мы получаем *приближенное* значение корня, можно для его вычисления использовать *приближенное* значение производной:

$$f'(x)=\lim_{\Delta x\to 0}\frac{f(x+\Delta x)-f(x)}{\Delta x}\approx\frac{f(x+\Delta x)-f(x)}{\Delta x}.$$

В качестве малой величины Δx можно взять, например, заданную точность вычислений ε, тогда расчетная формула примет вид

$$x_{k+1}=x_k-\frac{f(x_k)\varepsilon}{f(x_k+\varepsilon)-f(x_k)}. \qquad (0.1)$$

С другой стороны, для вычисления производной можно воспользоваться значениями функции, полученными на двух предыдущих шагах,

$$x_{k+1} = x_k - \frac{f(x_k)(x_k - x_k)}{f(x_k) - f(x_{k-1})} = \frac{f(x_k)x_{k-1} - f(x_{k-1})x_k}{f(x_k) - f(x_{k-1})}. \qquad (0.2)$$

В таком виде метод называется ***методом секущих*** (***secant method***). При этом, однако, возникает проблема с вычислением первого приближения. Обычно полагают, что $x_1 = x_0 + \varepsilon$, то есть первый шаг вычислений проводится с использованием формулы (0.1), а все последующие – с спользованием формулы (0.2). Именно эта вычислительная схема реализована в пакете ***MathCAD***. Используя метод секущих, мы не можем гарантировать, что корень находится между двумя последними приближениями. Можно, однако, для вычисления очередного приближения использовать границы интервала, на котором функция меняет знак. Такой метод называется ***методом хорд*** (***false position method***).

Идея метода секущих развивается в ***методе Мюллера***. Однако в этом методе для нахождения очередного приближения используются три предыдущие точки. Иными словами, метод использует не линейную, а квадратичную интерполяцию функции. Расчетные формулы метода следующие[1]:

$$\begin{aligned} q &= \frac{x_k - x_{k-1}}{x_{k-1} - x_{k-2}}; \\ A &= qf(x_k) - q(1+q)f(x_{k-1}) + q^2 f(x_{k-2}); \\ B &= (2q+1)f(x_k) - (1+q)^2 f(x_{k-1}) + q^2 f(x_{k-2}); \\ C &= (1+q)f(x_k); \end{aligned} \qquad (0.3)$$

$$x_{k+1} = x_k - (x_k - x_{k-1})\left(\frac{2C}{B \pm \sqrt{B^2 - 4AC}}\right). \qquad (0.4)$$

Знак перед корнем выбирается так, чтобы абсолютное значение знаменателя было максимальным.

Поскольку поиск корня заканчивается, когда выполнится условие $|f(x_{k+1})| \le \varepsilon$, то возможно появление ложных корней. Например, для уравнения $x^2 + 0{,}0001 = 0$ ложный корень $x = 0$ появится в том случае, если точность поиска задана меньше, чем 0,0001. Увеличивая точность поиска, можно избавиться от ложных корней. Однако не для всех урав-

[1] Получите эти формулы самостоятельно по аналогии с методом Ньютона, оставив в разложении Тейлора первые три слагаемых.

нений такой подход работает. Например, для уравнения $\frac{1}{x}=0$, которое, очевидно, не имеет действительных корней, для любой, сколь угодно малой точности найдется значение x, удовлетворяющее критерию окончания поиска. Приведенные примеры показывают, что к результатам компьютерных вычислений всегда нужно относиться критически, анализировать их на правдоподобность. Чтобы избежать «подводных камней» при использовании любого стандартного пакета, реализующего численные методы, нужно иметь хотя бы минимальное представление о том, какой именно численный метод реализован для решения той или иной задачи.

В том случае, когда известен интервал, на котором расположен корень, можно воспользоваться иными методами нахождения решения уравнения.

В ***методе Риддера*** **(*Ridder's method*)** вычисляют значение функции в середине интервала $x_3=(x_2-x_1)/2$. Затем ищут экспоненциальную функцию такую, что $f(x_1)-2f(x_3)e^{Q}+f(x_2)e^{2Q}=0$. Затем применяют метод хорд, используя значения $f(x_1), f(x_3)e^{Q}, f(x_2)e^{2Q}$. Очередное значение вычисляют по формуле

$$x_4=x_3+(x_3-x_1)\frac{\operatorname{sign}\left(f(x_1)-f(x_2)\right)f(x_3)}{\sqrt{f(x_3)^2-f(x_1)f(x_2)}}. \qquad (0.5)$$

Метод Брента **(*Brent method*)** соединяет быстроту метода Риддера и гарантированную сходимость метода деления отрезка пополам. Метод использует обратную квадратичную интерполяцию, то есть ищет x как квадратичную функцию y. На каждом шаге проверяется локализация корня. Формулы метода достаточно громоздки и мы не будем их приводить.

Особые методы применяют для поиска корней полинома. В этом случае могут быть найдены все корни. После того как один из корней полинома найден, степень полинома может быть понижена, после чего поиск корня повторяется.

Метод Лагерра **(*Laguerre's method*)** основывается на следующих соотношениях для полиномов:

$$P_n(x)=(x-x_1)(x-x_2)\ldots(x-x_n);$$
$$\ln|P_n(x)|=\ln|x-x_1|+\ln|x-x_2|\ldots\ln|x-x_n|;$$

$$\frac{d\ln|P_n(x)|}{dx}=\frac{1}{x-x_1}+\frac{1}{x-x_2}\dots\frac{1}{x-x_n}=\frac{P_n'}{P_n}\equiv G;$$

$$-\frac{d^2\ln|P_n(x)|}{dx^2}=\frac{1}{(x-x_1)^2}+\frac{1}{(x-x_2)^2}\dots\frac{1}{(x-x_n)^2}=\left(\frac{P_n'}{P_n}\right)^2-\frac{P_n''}{P_n}\equiv H.$$

Предполагают, что корень x_1 находится на расстоянии a от текущего приближения, в то время как все другие корни находятся на расстоянии b: $x-x_1=a$; $x-x_i=b$, если $i=2,\ 3,\dots,n$. Тогда

$$\frac{1}{a}+\frac{n-1}{b}=G;$$

$$\frac{1}{a^2}+\frac{n-1}{b^2}=H;$$

$$a=\frac{n}{G\pm\sqrt{(n-1)(nH-G^2)}}.$$

Знак перед корнем выбирают с таким расчетом, чтобы получить наибольшее значение знаменателя.

Еще один метод, который применяют для поиска корней полиномов, – ***метод сопровождающей матрицы*** (***companion matrix***). Можно доказать, что матрица

$$\mathbf{A}=\begin{pmatrix}-\frac{a_{n-1}}{a_n} & -\frac{a_{n-2}}{a_n} & \dots & -\frac{a_1}{a_n} & -\frac{a_0}{a_n}\\ 1 & 0 & \cdots & 0 & 0\\ 0 & 1 & \cdots & 0 & 0\\ \vdots & & & & \vdots\\ 0 & 0 & \cdots & 1 & 0\end{pmatrix},$$

называемая сопровождающей матрицей для полинома $P(x)=\sum_{i=0}^{n}a_ix^i$, имеет собственные значения, равные корням полинома. Напомним, что собственными значениями матрицы называются такие числа, для которых выполняется равенство $\mathbf{A}x=\lambda x$ или $P(x)=\det(\mathbf{A}-\lambda\mathbf{I})=0$. Существуют весьма эффективные методы поиска собственных значений, о некоторых из них мы будем говорить далее. Таким образом, задачу поиска корней полинома можно свести к задаче поиска собственных значений сопровождающей матрицы.

1.1. Функции произвольного вида

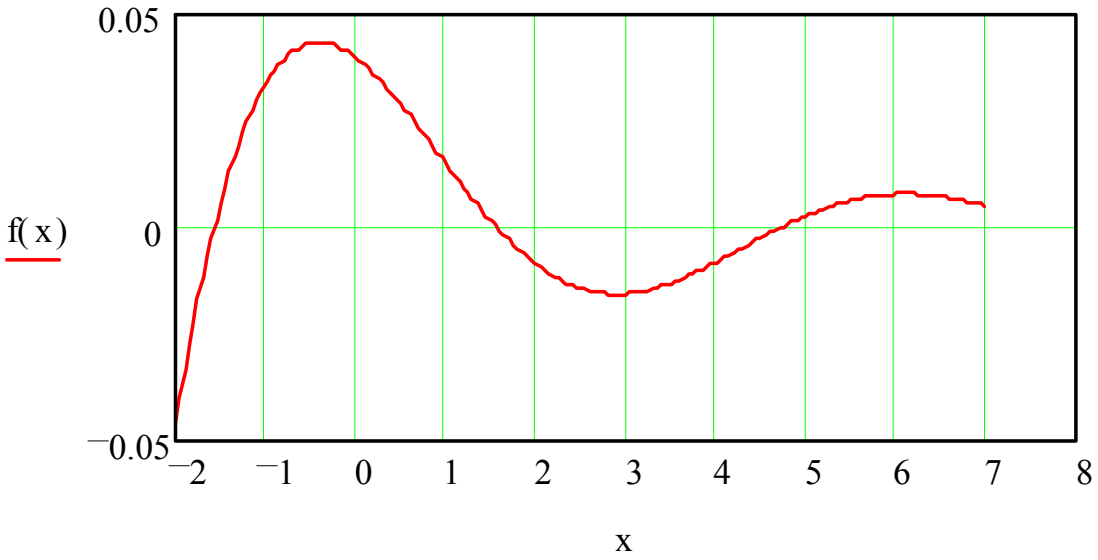
Найдем нули функции

$$f(x) := \frac{\cos(x)}{(x+5)^2}$$

на интервале x = [–2, 7], используя ***MathCAD***.

Изобразим сначала функцию на графике:

x := –2, –1.95 .. 7



На заданном интервале функция три раза обращается в нуль. Определим нули функции, используя встроенную функцию ***root*** **(*f*(*x*), *x*).** Первый аргумент – функция, нуль которой необходимо найти, ***второй*** – переменная, которую необходимо варьировать. (Вообще говоря, функция *f* может быть функцией многих переменных и необходимо указывать, по какой именно переменной мы ищем нуль функции.) Кроме того, необходимо задать начальное приближение поиска. Точность вычислений задается встроенной переменной TOL. По умолчанию ее значение равно 0.001. Это значение можно изменить либо через меню ***Math/Built–In Variables*** или непосредственно в тексте документа: $TOL := 10^{-9}$.

Задаем начальное приближение: x := –1. Вычисляем корень

root (f (x) , x) = –1.57079632

Если требуется найти несколько корней, как в нашей задаче, то имеет смысл определить новую функцию:

$r(x) := root(f(x), x)$

Функция `r(x)` возвращает ближайшее к x^2 значение корня, то есть начальное приближение мы задаем через аргумент функции. Задаем вектор начальных приближений `x` и находим соответствующие им корни `X`:

$i := 0..2$ $\qquad X_i := r(x_i)$

$x_i :=$

-1
1
4

X_i

-1.570796327
1.570796327
4.712388981

Для данного примера корни легко могут быть найдены аналитически. Полученный численный результат с заданной точностью совпадает с точным решением.

Определение новой функции целесообразно и в том случае, когда мы хотим исследовать зависимость решения от параметра. Пусть функция зависит от параметра `a`:

$$z(a, x) := root\left[\frac{\cos(a \cdot x)}{(x+5)^2}, x\right]$$

Первый аргумент функции `z` задает значение параметра, второй – начальное приближение. Найдем корни уравнения при значениях параметра 1 и 2:

$z(1, 1) = 1.571$

$z(2, 1) = 0.785$

Если мы хотим получить комплексный корень, то начальное приближение следует задавать комплексным:

$z(i, i) = -7.459 \cdot 10^{-9} + 1.571i$

[2] К сожалению, это не всегда так. Если начальное приближение выбрано неудачно и значение производной в этой точке близко к нулю, то, вообще говоря, найденный корень может быть не ближайшим к начальному приближению. В качестве примера решите самостоятельно задачу поиска корня уравнения $\sin(x)=0$, выбрав в качестве начального приближения число, близкое к $\frac{\pi}{2}$. Чем ближе к $\frac{\pi}{2}$ будет выбранное значение, тем более далекий от 0 корень мы будем получать.

1.2. Нахождение корней полиномов

Для нахождения корней полиномов имеется встроенная функция ***polyroots(a)***. Аргументом функции является вектор коэффициентов полинома $a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$, то есть для уравнения $3x^2 + 2x + 1 = 0$ вектор a имеет вид

$$i := 0 .. 2 \qquad a_i := \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \qquad \text{polyroots}(a) = \begin{pmatrix} -0.333 - 0.471i \\ -0.333 + 0.471i \end{pmatrix}$$

Если в полиноме отсутствуют некоторые степени, то на соответствующих местах следует писать 0. Пусть требуется найти корни полинома $p(x) = x^3 + 2x^2 - 1$

$$i := 0 .. 3 \qquad a_i := \begin{bmatrix} -1 \\ 0 \\ 2 \\ 1 \end{bmatrix} \qquad \text{polyroots}(a) = \begin{pmatrix} -1.618 \\ -1 \\ 0.618 \end{pmatrix}$$

Коэффициенты полинома могут быть и комплексными.

1.3. Нахождение корней уравнений путем символических преобразований

Во многих случаях ***MathCAD*** позволяет найти аналитическое решение уравнения. Для этого необходимо воспользоваться пунктом ***Solve for Variable*** из пункта меню ***Symbolic***. Для того чтобы найти решение уравнения необходимо записать выражение и выделить в нем переменную (поставить указатель курсора возле переменной). Это необходимо для того, чтобы показать, какая именно величина является переменной, а какая – фиксированным параметром. После этого выбираем из пункта меню ***Symbolic*** подпункт ***Solve for Variable***:

$$\frac{\cos(a \cdot x)}{(x + 5)^2}$$

Решение готово: $\frac{1}{2}\cdot\frac{\pi}{a}$

***Обратите внимание*!** В данном случае был найден только один корень, хотя, очевидно, их бесконечно много.

В случае полинома ***MathCAD***, а точнее – встроенный символический процессор Maple, – находит все корни.

Для примера $x^2 + 2\cdot x + 1 \rightarrow \begin{pmatrix} -1 \\ -1 \end{pmatrix}$ найдено 2 корня, хотя они и вырождены. Пример с комплексными корнями: $x^2 + 1 \longrightarrow \begin{pmatrix} i \\ -i \end{pmatrix}$.

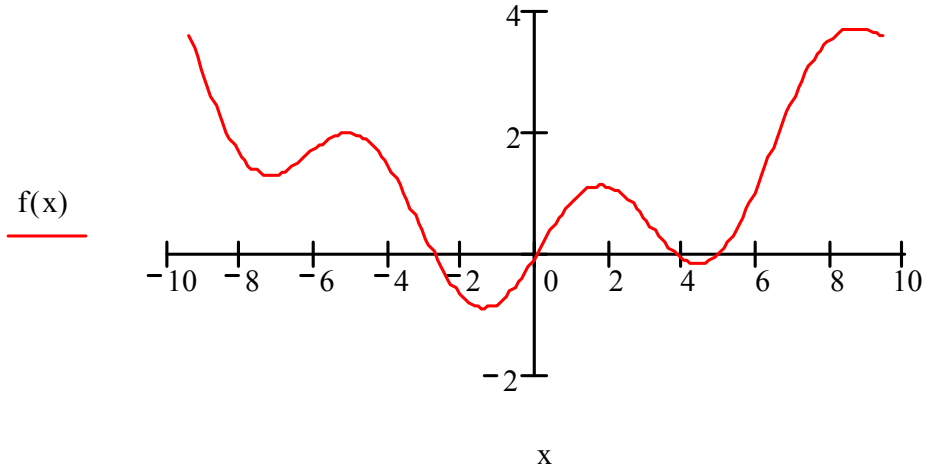
1.4. Поиск корней уравнений в MathCAD

MathCAD представляет ряд дополнительных возможностей для поиска корней уравнений. Функция **root(*f* (*var*1, *var*2, ...), *var*1, [*a*, *b*])** имеет теперь два необязательных аргумента `a` и `b`, которые определяют границы интервала, на котором следует искать корень. На концах интервала `[a, b]` функция `f` должна менять знак **(*f*(*a*)*f*(*b*) < 0)**. Задавать начальное приближение для корня не нужно. В данном варианте функция root использует алгоритм Риддера и Брента. Продемонстрируем использование расширенного варианта поиска корней на примере функции

$$f(x) := 0.04 \cdot x^2 + \sin(x)$$

Для оценки местоположения корней построим график этой функции:

$$x := -3\cdot\pi, -3\cdot\pi + 0.1 .. 3\cdot\pi .$$



$\text{root}(f(x), x, -1, 8) = 0$ $\quad$ $\text{root}(f(x), x, -10, -0.1) = -2.818$

На интервале [1, 8] функция имеет два корня. ***MathCAD*** смог найти только один из них.

Дополнительные возможности появились и для нахождения корней полиномов. Функция ***polyroots*** может использовать два различных алгоритма поиска корней – метод Лагерра и метод сопровождающей матрицы. Переключение методов осуществляется в контекстном меню, которое вызывается нажатием правой кнопки мыши, когда указатель установлен на имя функции.

2. РЕШЕНИЕ СИСТЕМ УРАВНЕНИЙ И НЕРАВЕНСТВ

Формально задача поиска решения системы уравнений

$$f_1(x_1, x_2,\ldots,x_n)=0;$$
$$f_2(x_1, x_2,\ldots,x_n)=0,$$
$$\ldots$$
$$f_n(x_1, x_2,\ldots,x_n)=0$$

может быть записана точно так же, как и задача поиска корня одного уравнения $\mathrm{f}(\mathrm{x})=0$, где $\mathrm{f}=(f_1, f_2,\ldots,f_n)$ и $\mathrm{x}=(x_1, x_2,\ldots,x_n)$. Вблизи точки x каждая из функций f_i может быть разложена в ряд Тейлора

$$f_i(\mathrm{x}+\Delta\mathrm{x})=f_i(\mathrm{x})+\sum_{j=1}^{n}\frac{\partial f_i}{\partial x_j}\Delta x_j+\cdots$$

или в векторной форме

$$\mathrm{f}(\mathrm{x}+\Delta\mathrm{x})=\mathrm{f}(\mathrm{x})+\mathrm{J}\,\Delta\mathrm{x}+\cdots,$$

где J – матрица Якоби с элементами

$$J_{i,j}=\frac{\partial f_i}{\partial x_j}.$$

Ограничиваясь только первыми двумя членами разложения и полагая, что $\mathrm{f}(\mathrm{x}+\Delta\mathrm{x})=0$, получаем уравнение $\Delta\mathrm{x}=-\mathrm{J}^{-1}\mathrm{f}(\mathrm{x})$.

Таким образом, мы получаем схему для уточнения решения системы уравнений, аналогичную методу Ньютона для случая одного уравнения:

$$\mathrm{x}^{(\mathrm{k}+1)}=\mathrm{x}^{(\mathrm{k})}-\mathrm{J}^{-1}\mathrm{f}(\mathrm{x}^{(\mathrm{k})}). \qquad (0.6)$$

Поскольку вычислять матрицу Якоби на каждом шаге достаточно трудоемко, то обычно ее элементы вычисляют приближенно или используют одни и те же значения на нескольких шагах.

Одну из разновидностей метода Ньютона использует ***MathCAD***.

2.1. Решение систем линейных и нелинейных уравнений и неравенств

Системы линейных и нелинейных уравнений и неравенств позволяет решать на ***MathCAD*** блок ***given*** в сочетании с функцией ***Find***.

Внимание*!** В блоке ***given записывается система уравнений и/или неравенств, подлежащих решению.

Система уравнений и/или неравенств должна быть записана после или правее слова ***given***.

При записи уравнений вместо знака = следует набирать ***Ctrl***+=.

Перед словом ***given*** необходимо указывать начальные приближения для всех переменных.

Блок ***given*** не пригоден для поиска индексированных переменных.

Если мы хотим найти комплексный корень, следует задавать комплексное начальное приближение.

Признаком окончания системы служит функция ***Find***, если мы хотим найти точное решение системы, либо функция ***Minerr***, если система не может быть решена точно, и мы хотим найти наилучшее приближение, обеспечивающее минимальную погрешность.

Функции ***Minerr*** и ***Find*** должны иметь столько же или меньше аргументов, сколько уравнений и неравенств содержит блок ***given***. Если окажется, что блок содержит слишком мало уравнений или неравенств, то его можно дополнить тождествами или повторяющимися выражениями.

В том случае, если решение не может быть найдено при заданном выборе начального приближения, появится сообщение в красной рамке ***Did not find solution*** – решение не найдено.

Зададим начальные приближения и решим систему нелинейных уравнений:

$x := 1 \quad y := 1$

given $\quad x^2 - y = 23 \quad x^2 \cdot y = 50 \quad$ Find $(x, y) = \begin{pmatrix} 5 \\ 2 \end{pmatrix}$

Если необходимо найти решение при различных начальных приближениях, имеет смысл определить новую функцию

given $\quad x^2 - y = 23 \quad x^2 \cdot y = 50$

$f(x, y) := \text{Find}(x, y)$

Обратите внимание*!** В этом случае не нужно задавать начальные приближения перед началом блока ***given – Find. Начальные приближения задаются в качестве аргументов функции `f(x, y)`

$$f(1,1)=\begin{pmatrix}5\\2\end{pmatrix} \qquad f(-1,1)=\begin{pmatrix}-5\\2\end{pmatrix}$$

$$f(i,1)=\begin{pmatrix}1.414i\\-25\end{pmatrix} \qquad f(-i,1)=\begin{pmatrix}-1.414i\\-25\end{pmatrix}$$

Подобным же образом можно решать системы, зависящие от параметра:

given $x^2+y^2=R^2$ $y=x$

$$g(1,1,1)=\begin{pmatrix}0.707\\0.707\end{pmatrix} \qquad g(-1,-1,1)=\begin{pmatrix}-0.707\\-0.707\end{pmatrix}$$

$$g(1,1,2)=\begin{pmatrix}1.414\\1.414\end{pmatrix} \qquad g(-1,-1,2)=\begin{pmatrix}-1.414\\-1.414\end{pmatrix}$$

2.2. Решение систем линейных уравнений и неравенств

Совершенно аналогично решаются системы линейных уравнений.

Однако в том случае, когда система линейных уравнений невырождена, то есть ее определитель отличен от нуля, более изящным (хотя и не самым эффективным с точки зрения вычислительной математики) является матричный способ решения.

Решим линейную систему одним и другим методом:

$x := -2$ $y := 1$ $z := 2$

given $2\cdot x-4\cdot y+3\cdot z=1$

$x-2\cdot y+4\cdot z=3$ $3\cdot x-y+5\cdot z=2$

$$\text{Find}(x,y,z)=\begin{pmatrix}-1\\-3.904\cdot 10^{-10}\\1\end{pmatrix}$$

Решим ту же самую систему матричным методом.

Запишем матрицу системы и вектор свободных членов:

$$a:=\begin{pmatrix}2&-4&3\\1&-2&4\\3&-1&5\end{pmatrix} \qquad b:=\begin{pmatrix}1\\3\\2\end{pmatrix}$$

Решим систему, умножая слева столбец свободных членов `b` на матрицу, обратную матрице `a`:

$$x := a^{-1} \cdot b \qquad x = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}$$

Если вы работаете с продвинутой версией ***MathCAD***, то для этих же целей можно воспользоваться встроенной функцией ***lsolve***:

$$\text{lsolve}(a, b) = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}$$

2.3. Символическое решение систем уравнений

Во многих случаях решение системы уравнений может быть найдено не только численно, но и аналитически. Для этого так же используется блок ***given*** и функция ***Find***, но вместо знака равенства после функции следует поставить знак символического преобразования –> (Ctrl+.):

given

$x^2 + x \cdot y + y^2 = 13$

$x + y = 4$

$$\text{Find}(x, y) \rightarrow \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix}$$

Решение записано в виде матрицы. Каждый столбец соответствует паре (x, y), то есть найдены решения (1, 3) и (3, 1).

2.4. Нахождение экстремумов функций

Для нахождения экстремумов функций многих переменных существует две альтернативные возможности.

Первая заключается в использовании блока ***given*** и функции ***minerr***.

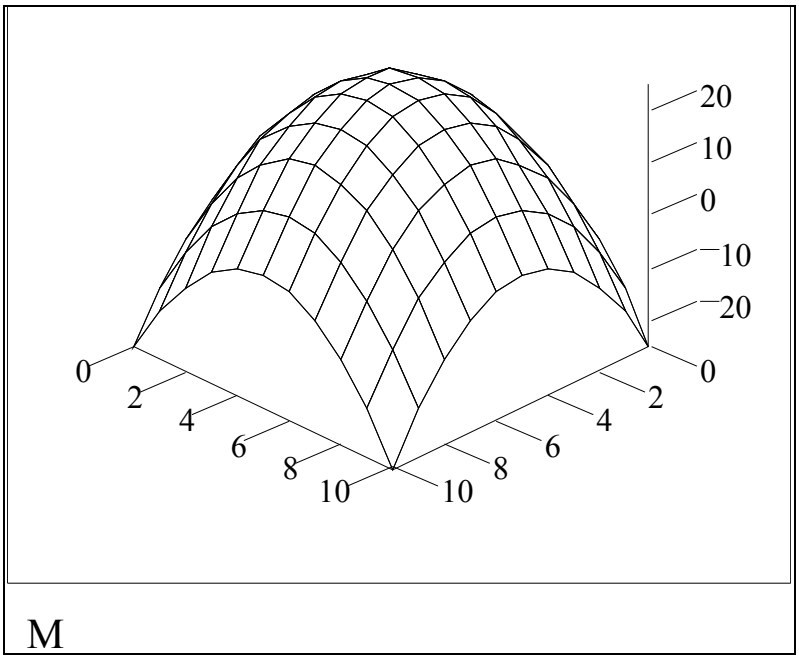
Определим функцию двух переменных

$$G(x, y) := 25 - x^2 - y^2$$

Зададимся целью найти ее экстремум в области x = [–5, 5]; y = [–5, 5]. Оценим по графику положение экстремума.

Заносим в матрицу M значения функции в узловых точках:

$$i := 0 \,..\, 10 \qquad j := 0 \,..\, 10 \qquad M_{(i,j)} := G(i - 5, j - 5)$$



На заданном интервале функция не превосходит 25. Зададим начальные приближения для поиска экстремума:

$x := 1 \qquad y := 1$

Записываем блок уравнений или неравенств. Число уравнений и неравенств в блоке ***given – Find*** должно быть больше и равно числа искомых величин. Если уравнений и неравенств не хватает, то можно просто продублировать одно и то же уравнение или вписать какое-либо тождество, например, 2 = 2:

given $\quad G(x,y) = 26 \qquad x \geq -5 \qquad x \leq 5 \quad y \geq -5 \qquad y \leq 5$

$$\text{Minerr}(x,y) = \begin{pmatrix} 4.584 \cdot 10^{-6} \\ 4.02 \cdot 10^{-5} \end{pmatrix}$$

Функция ***Minerr*** ищет приближенной решение для системы уравнений и неравенств, записанных в блоке. В данном случае мы получили, что системе уравнений наилучшим образом соответствует точка [0, 0]. (Поскольку по умолчанию точность вычислений составляет 0.001, мы округлили результат до 0.)

Из графика видно, что значение 26 больше самого большого значения функции в окрестностях точки [1, 1], то есть точное решение найти нельзя и функция ***Minerr*** подбирает такое значение x, при котором функция ближе всего к значению 26.

Вторая возможность – поиск нулей первой производной, то есть стандартный математический подход. Для этого можно использовать блок ***given – Find***. Функция ***Find*** ищет точное решение системы уравнений и неравенств, записанных после слова ***given***:

$$\text{Given} \quad \frac{d}{dx}G(x,y)=0 \qquad \frac{d}{dy}G(x,y)=0$$

$$\text{Find}(x,y)=\begin{pmatrix} 1.059\cdot 10^{-7} \\ 1.059\cdot 10^{-7} \end{pmatrix}$$

Результаты, полученные различными методами, совпадают, время счета мало в обоих случаях.

В старших версиях ***MathCAD*** появилась дополнительная возможность поиска экстремумов с помощью функций ***Minimize*** и ***Maximize***, которые могут быть использованы как сами по себе, так и совместно с блоком given. Аргументы функций: имя функции, экстремум которой ищется, и список ее аргументов.

Определяем функцию двух переменных $f(x,y) := 25 - x^2 - y^2$ и задаем начальные приближения $x := 1 \qquad y := 1$.

Задаем область поиска максимума внутри блока ***given***:

Given

$0 \le x \le 5$

$0 \le y \le 5$

Находим максимум функции в заданной области

$P := \text{Maximize}(f, x, y)$

$$P=\begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

В случае функции одной переменной задаем функцию $g(x) := x^4 - x^2$ и начальное приближение $x := 1$.

Находим максимум $\text{Maximize}(g, x) = 0$.

Если мы хотим найти максимальное или минимальное значение функции на некотором интервале, то необходимо определить этот интервал в блоке given:

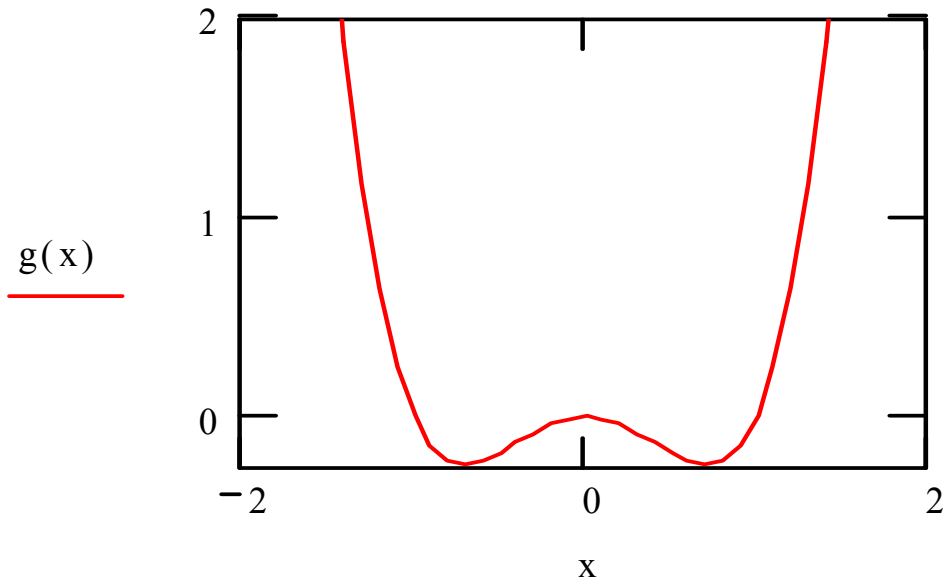
Given

$0 < x \le 2$

Maximize (g , x) = 2

На приведенном ниже графике видно, что первый из найденных максимумов соответствовал случаю, когда производная обращается в нуль; второй максимум лежит на границе интервала:

x := – 2 , – 1.9 .. 2



3. АППРОКСИМАЦИЯ ФУНКЦИЙ

Аппроксимацией (приближением) функции $f(x)$ называется нахождение такой функции $g(x)$ (***аппроксимирующей функции***), которая была бы близка заданной. Критерии близости функций $f(x)$ и $g(x)$ могут быть различные.

В том случае, когда приближение строится на дискретном наборе точек, аппроксимацию называют ***точечной*** или ***дискретной***.

В том случае, когда аппроксимация проводится на непрерывном множестве точек (отрезке), аппроксимация называется ***непрерывной*** или ***интегральной***. Примером такой аппроксимации может служить разложение функции в ряд Тейлора, то есть замена некоторой функции степенным многочленом.

Наиболее часто встречающим видом точечной аппроксимации является ***интерполяция*** (в широком смысле).

Пусть задан дискретный набор точек x_i $(i=0,\ 1,...,n)$, называемых ***узлами интерполяции***, причем среди этих точек нет совпадающих, а также значения функции y_i в этих точках. Требуется построить функцию $g(x)$, проходящую через все заданные узлы. Таким образом, критерием близости функции является

$$g(x_i)=y_i.$$

В качестве функции $g(x)$ обычно выбирается полином, который называют ***интерполяционным полиномом***.

В том случае, когда полином един для всей области интерполяции, говорят, что интерполяция ***глобальная***.

В тех случаях, когда между различными узлами полиномы различны, говорят о ***кусочной*** или ***локальной интерполяции.***

Найдя интерполяционный полином, мы можем вычислить значения функции $f(x)$ между узлами (провести ***интерполяцию в узком смысле слова***), а также определить значение функции $f(x)$ даже за пределами заданного интервала (провести ***экстраполяцию***).

Следует иметь в виду, что точность экстраполяции обычно очень невелика.

3.1. Локальная интерполяция

3.1.1. Линейная интерполяция

Простейшим случаем локальной интерполяции является линейная интерполяция, когда в качестве интерполяционной функции выбирается полином первой степени, то есть узловые точки соединяются прямой линией.

Линейная интерполяция на ***MathCAD*** осуществляется с помощью встроенной функции ***linterp***.

Пусть требуется провести линейную интерполяцию функции sin(*x*) на интервале [0…6], используя пять узлов интерполяции, и вычислить значения функции в четырех точках X_k:

$k := 0 .. 3$

$X_k :=$

-0.5
1.111
2.333
4.574

Задаем интервал изменения *x* и число узловых точек

$x_{min} := 0$ $x_{max} := 6$ $n := 5$

Определяем шаг изменения x: $h := \frac{(x_{max} - x_{min})}{n}$

Вычисляем координаты узлов и значения функции в них:

$i := 0 .. n - 1$ $x_i := x_{min} + i \cdot h$ $y_i := \sin(x_i)$

Проводим линейную интерполяцию: $g(z) := linterp(x, y, z)$

Вычислим значение интерполяционной функции в заданных точках и сравним их с точными значениями:

$g(X_k)$
-0.388
0.863
0.69
-0.892

$\sin(X_k)$
-0.479
0.896
0.723
-0.99

Как видно, результаты интерполяции отличаются от точных значений функции незначительно.

3.1.2. Интерполяция сплайнами

В настоящее время среди методов локальной интерполяции наибольшее распространение получила интерполяция сплайнами (от англ. ***spline*** – гибкая линейка). При этом строится интерполяционный полином третьей степени, проходящий через все заданные узлы и имеющий непрерывные первую и вторую производные. На каждом интервале $[x_i,\ \ x_{i+1}]$ интерполирующая функция является полиномом третьей степени

$$S(x) = S_i(x) = a_0^{(i)} + a_1^{(i)}(x - x_i) + a_2^{(i)}(x - x_i)^2 + a_3^{(i)}(x - x_i)^3$$

и удовлетворяет условиям

$$S(x_i) = y_i\,. \qquad (0.7)$$

Если всего n узлов, то интервалов – $n-1$. Значит, требуется определить $4(n-1)$ неизвестных коэффициентов полиномов. Условие (0.7) дает нам n уравнений. Условие непрерывности функции и ее первых двух производных во внутренних узлах интервала дает дополнительно $3(n-2)$ уравнений:

$$S_i(x_{i+1}) = S_{i+1}(x_{i+1});$$
$$S_i'(x_{i+1}) = S_{i+1}'(x_{i+1});$$
$$S_i''(x_{i+1}) = S_{i+1}''(x_{i+1}).$$

Всего имеем $4n-6$ различных уравнений. Два недостающих уравнения можно получить, задавая условия на краях интервала. В частности, можно потребовать нулевой кривизны функции на краях интервала, то есть $S''(a) = S''(b) = 0$.

Задавая различные условия на концах интервала, можно получить разные сплайны.

Решим задачу об интерполяции синуса с помощью сплайнов. Для этого воспользуемся встроенной функцией ***interp*(*VS*, *x*, *y*, *z*)**. Переменные ***x*** и ***y*** задают координаты узловых точек, ***z*** является аргументом функции, ***VS*** определяет тип граничных условий на концах интервала.

Определим интерполяционные функции для трех типов кубического сплайна:

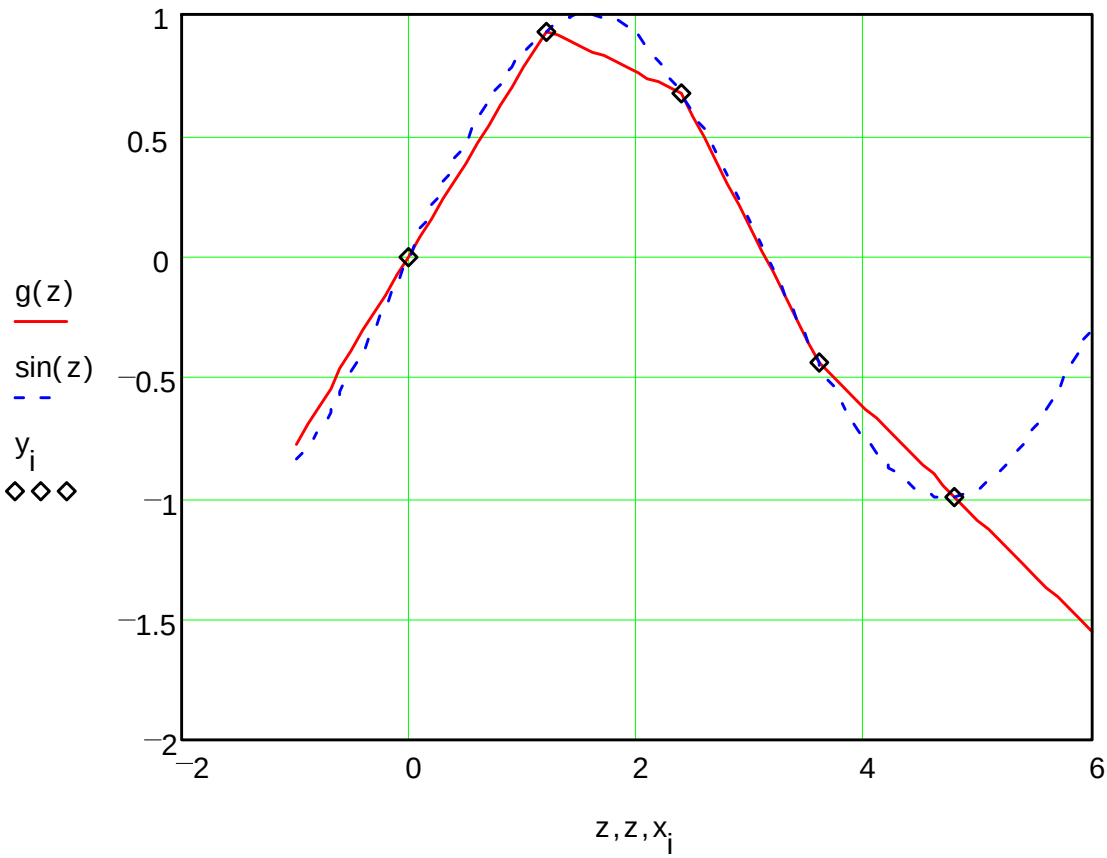
```
VSl := lspline(x, y)        gl(z) := interp(VSl, x, y, z)
VSp := pspline(x, y)        gp(z) := interp(VSp, x, y, z)
VSc := cspline(x, y)        gc(z) := interp(VSc, x, y, z)
```

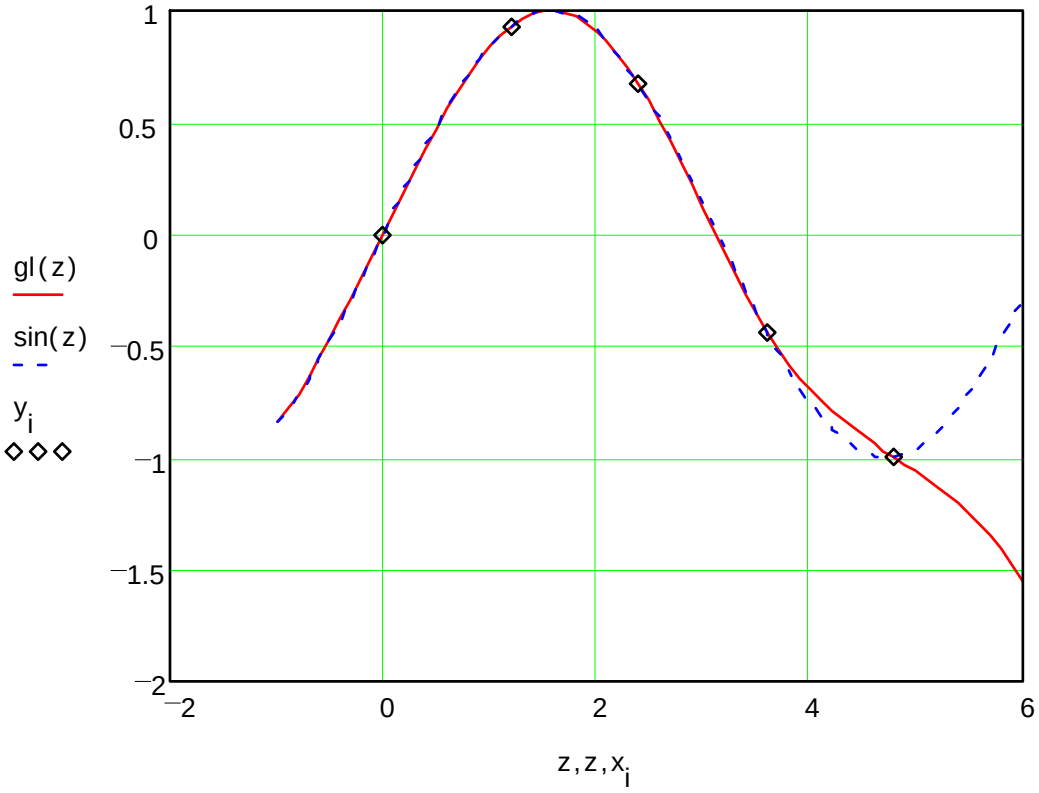
Вычисляем значения интерполяционных функций в заданных точках и сравниваем результаты с точными значениями:

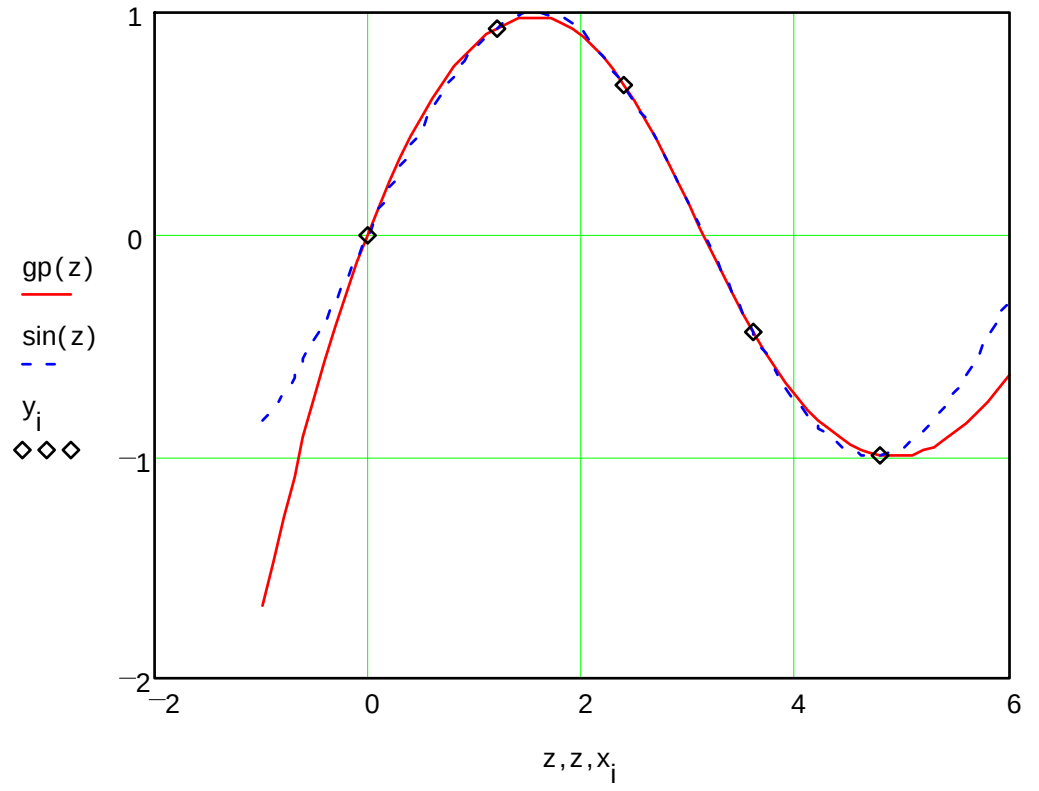
X_k	$gl(X_k)$	$gp(X_k)$	$gc(X_k)$	$\sin(X_k)$
-0.5	-0.473	-0.737	-0.752	-0.479
1.11	0.896	0.903	0.904	0.896
2.33	0.724	0.722	0.72	0.723
4.57	-0.927	-0.962	-1.014	-0.99

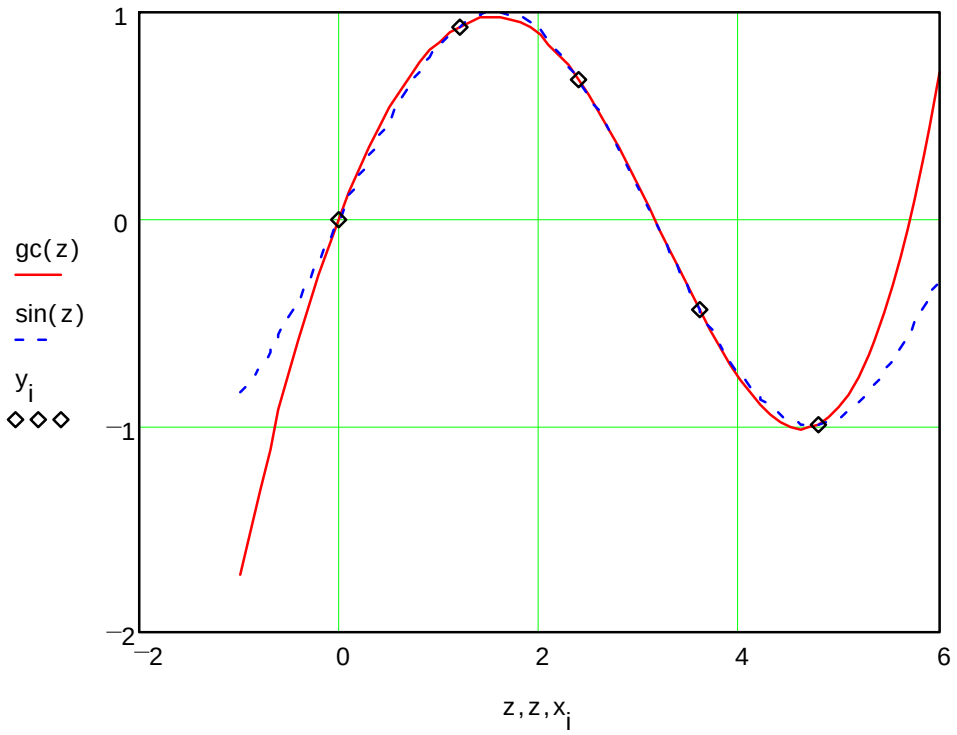
Обратите внимание, что результаты интерполяции различными типами кубических сплайнов практически не отличаются во внутренних точках интервала и совпадают с точными значениями функции. Вблизи краев интервала отличие становится более заметным, а при экстраполяции за пределы заданного интервала различные типы сплайнов дают существенно разные результаты. Для большей наглядности представим результаты на графиках

$z := -1, -0.9 .. 6$

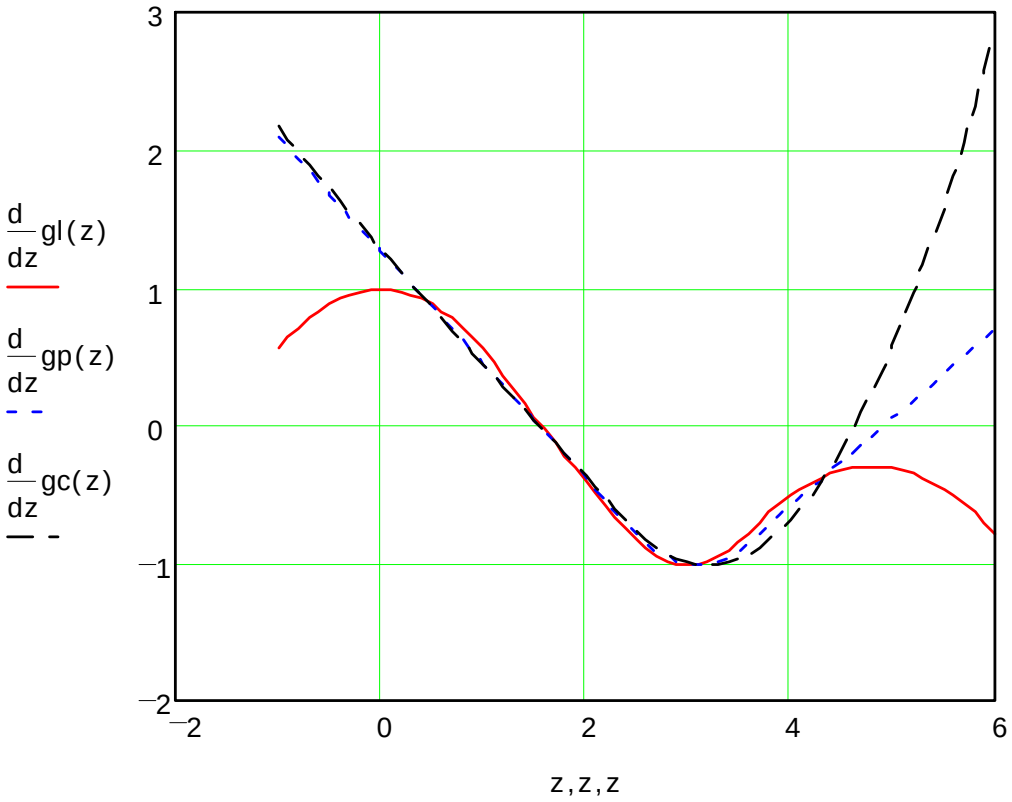


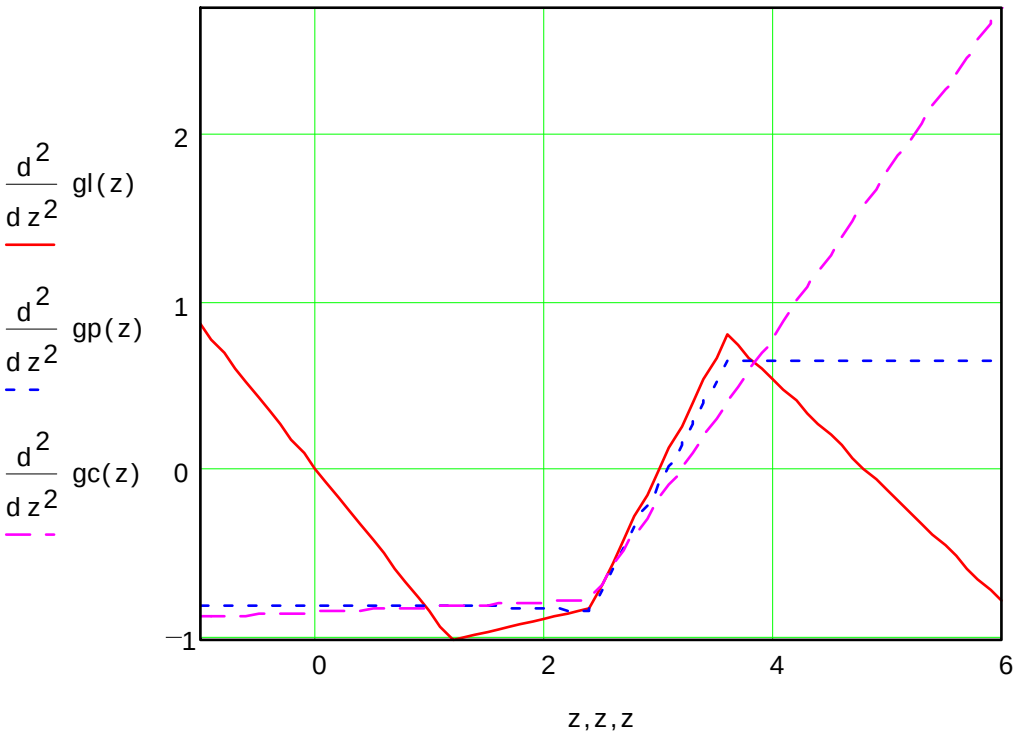

gl(z)
sin(z)
y_i
1
0.5
0
−0.5
−1
−1.5
−2
−2
0
2
4
6
z, z, x_i


gp(z)
sin(z)
y_i
1
0
−1
−2
−2
0
2
4
6
z, z, x_i



Убедимся в том, что первые и вторые производные сплайна непрерывны.





Но производные более высоких порядков уже не являются непрерывными.

3.2. Глобальная интерполяция

При глобальной интерполяции ищется единый полином для всего интервала. Если среди узлов $\{x_i, y_i\}$ нет совпадающих, то такой полином будет единственным, и его степень не будет превышать n.

Запишем систему уравнений для определения коэффициентов полинома $(n–1)$-й степени

$$c_0 + c_1 x_0 + c_2 x_0^2 + \ldots + c_{n-1} x_0^{n-1} = y_0;$$
$$c_0 + c_1 x_1 + c_2 x_1^2 + \ldots + c_{n-1} x_1^{n-1} = y_1;$$
$$\vdots$$
$$c_0 + c_1 x_1 + c_2 x_{n-1}^2 + \ldots + c_{n-1} x_{n-1}^{n-1} = y_{n-1}.$$

Определим матрицу коэффициентов системы уравнений

$i := 0 .. n - 1$

$j := 0 .. n - 1$

$a_{(j,i)} := (x_j)^i$

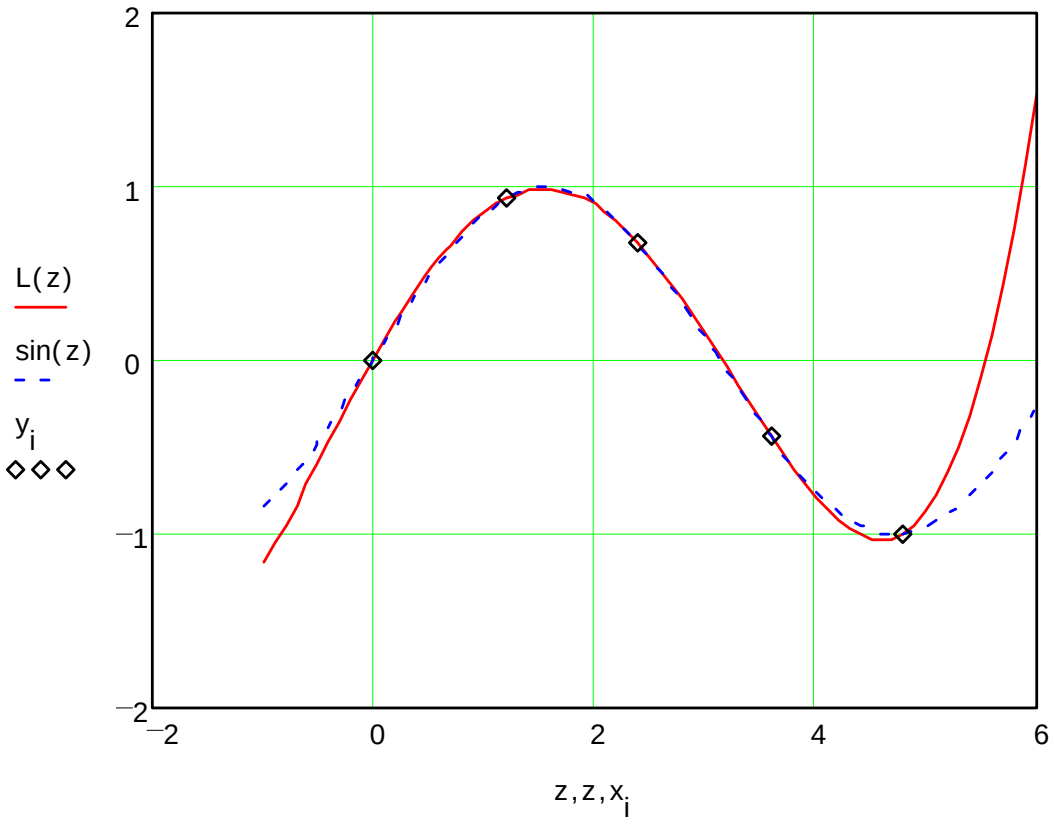
Решим систему уравнений матричным методом:

$$c := a^{-1} \cdot y$$

Вычислим интерполяционный полином

$$L(z) := \sum_i c_i \cdot z^i$$

Представим результаты на графике, например, для функции $y = \sin(x)$.



Вычислим значения интерполяционного полинома в заданных точках и сравним их с точными значениями:

X_k
-0.5
1.111
2.333
4.574

$L(X_k)$
-0.594
0.901
0.72
-1.036

$\sin(X_k)$
-0.479
0.896
0.723
-0.99

Коэффициенты интерполяционного полинома следующие:

$$c = \begin{bmatrix} 0 \\ 1.134 \\ -0.177 \\ -0.127 \\ 0.022 \end{bmatrix}$$

***Внимание*!** Из-за накопления вычислительной погрешности (ошибок округления) при большом числе узлов (n>10) возможно резкое ухудшение результатов интерполяции. Кроме того, для целого ряда функций глобальная интерполяция полиномом вообще не дает удовлетворительного результата. Рассмотрим в качестве примера две таких функции. Для этих функций точность интерполяции с ростом числа узлов не увеличивается, а уменьшается. Первым примером является функция

$$f(x) := \frac{1}{1 + 25 \cdot x^2}$$

Построим для нее интерполяционный полином на интервале [–1, 1], используя 9 точек:

$$n := 9 \qquad i := 0 .. n - 1 \qquad j := 0 .. n - 1$$

$$x_j := -1 + \frac{j}{n - 1} \cdot 2$$

$$a_{(j,i)} := (x_j)^i \qquad y_j := f(x_j) \qquad c := a^{-1} \cdot y \qquad L(z) := \sum_i c_i \cdot z^i$$

Представим результаты на графике:

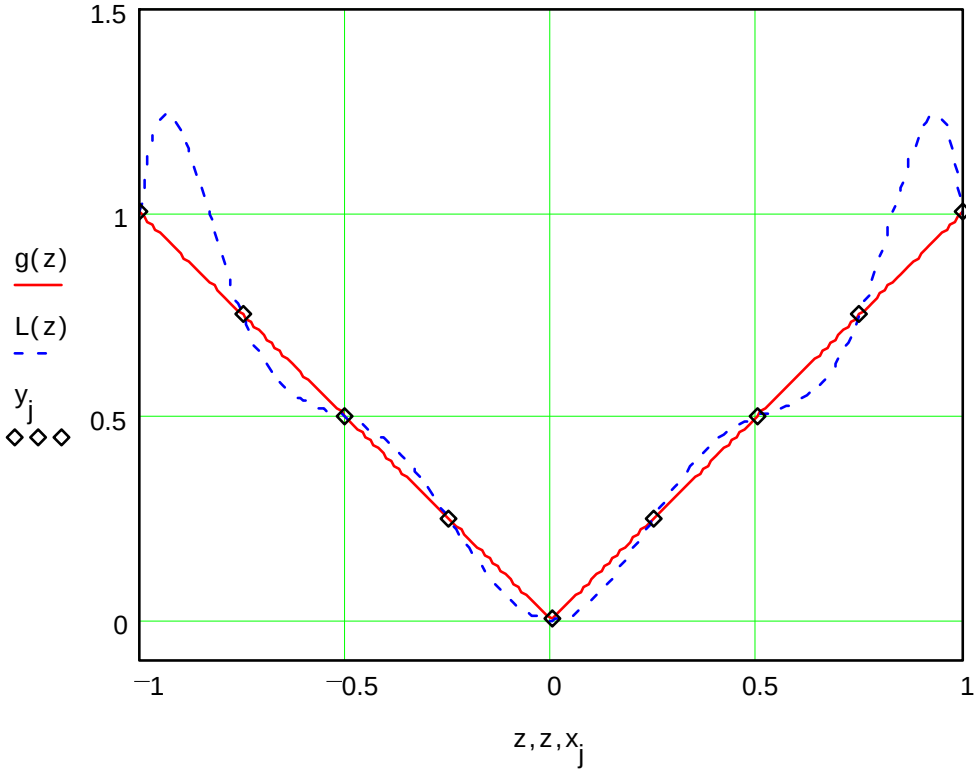
$$z := -1 , -0.99 .. 1$$

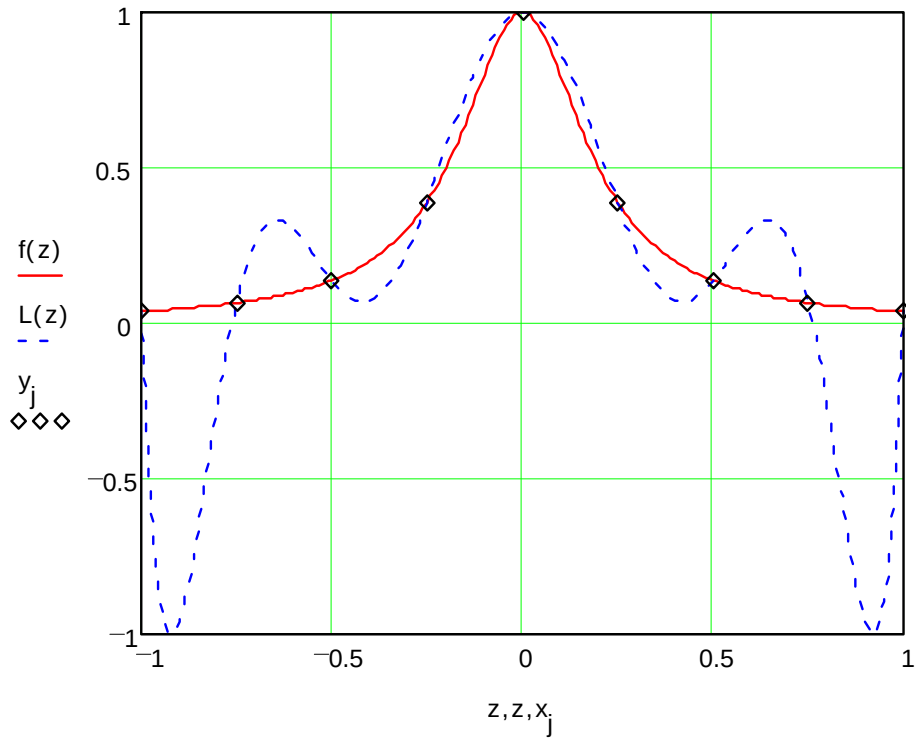
Второй пример – функция

$$g(x) := |x|$$

Найдем интерполяционный полином, используя заданные выше точки:

$$y_j := g(x_j) \qquad c := a^{-1} \cdot y \qquad L(z) := \sum_i c_i \cdot z^i$$





Убедитесь самостоятельно, что, при увеличении числа узлов интерполяции, результаты интерполирования вблизи концов интервала ухудшаются.

3.3. Метод наименьших квадратов

Наиболее распространенным методом аппроксимации экспериментальных данных является метод наименьших квадратов. Метод позволяет использовать аппроксимирующие функции произвольного вида и относится к группе глобальных методов. Простейшим вариантом метода наименьших квадратов является аппроксимация прямой линией (полиномом первой степени). Этот вариант метода наименьших квадратов носит также название линейной регрессии.

Критерием близости в методе наименьших квадратов является требование минимальности суммы квадратов отклонений от аппроксимирующей функции до экспериментальных точек:

$$\Phi = \sum_{i=1}^{n} \left(y_i - f(x_i) \right)^2 \to \min .$$

Таким образом, не требуется, чтобы аппроксимирующая функция проходила через все заданные точки, что особенно важно при аппроксимации данных, заведомо содержащих погрешности.

Важной особенностью метода является то, что аппроксимирующая функция может быть произвольной. Ее вид определяется особенностями решаемой задачи, например, физическими соображениями, если проводится аппроксимация результатов физического эксперимента. Наиболее часто встречаются аппроксимация прямой линией (линейная регрессия), аппроксимация полиномом (полиномиальная регрессия), аппроксимация линейной комбинацией произвольных функций. Кроме того, часто бывает возможно путем замены переменных свести задачу к линейной (провести линеаризацию). Например, пусть аппроксимирующая функция ищется в виде $y = A\exp(kx)$. Прологарифмируем это выражение и введем обозначения $z = \ln(y)$, $a = \ln(A)$. Тогда в новых обозначениях задача сводится к отысканию коэффициентов линейной функции $z = a + kx$.

3.3.1. Аппроксимация линейной функцией

Применим метод наименьших квадратов для аппроксимации экспериментальных данных.

Читаем данные из файлов `datax` и `datay`:

```
x := READPRN ( datax )        y := READPRN ( datay )
```

При использовании ***MathCAD*** имя файла следует заключать в кавычки и записывать его по правилам MS DOS, например, READPRN("c:\mylib\datax.prn").

Определяем количество прочитанных данных (число экспериментальных точек):

$n := \text{last}(x)$

$i := 0 .. n$

Используем встроенные функции ***slope*** и ***intercept*** для определения коэффициентов линейной регрессии (аппроксимация данных прямой линией). Функция ***slope*** определяет угловой коэффициент прямой, а функция ***intercep*** – точку пересечения графика с вертикальной осью:

$A := \text{intercept}(x, y)$

$B := \text{slope}(x, y)$

Определяем аппроксимирующую функцию:

$f1(z) := A + B \cdot z$

Коэффициенты линейной регрессии:

$A = -3.539$

$B = 1.81$

MathCAD предлагает для этих же целей использовать функцию ***line***

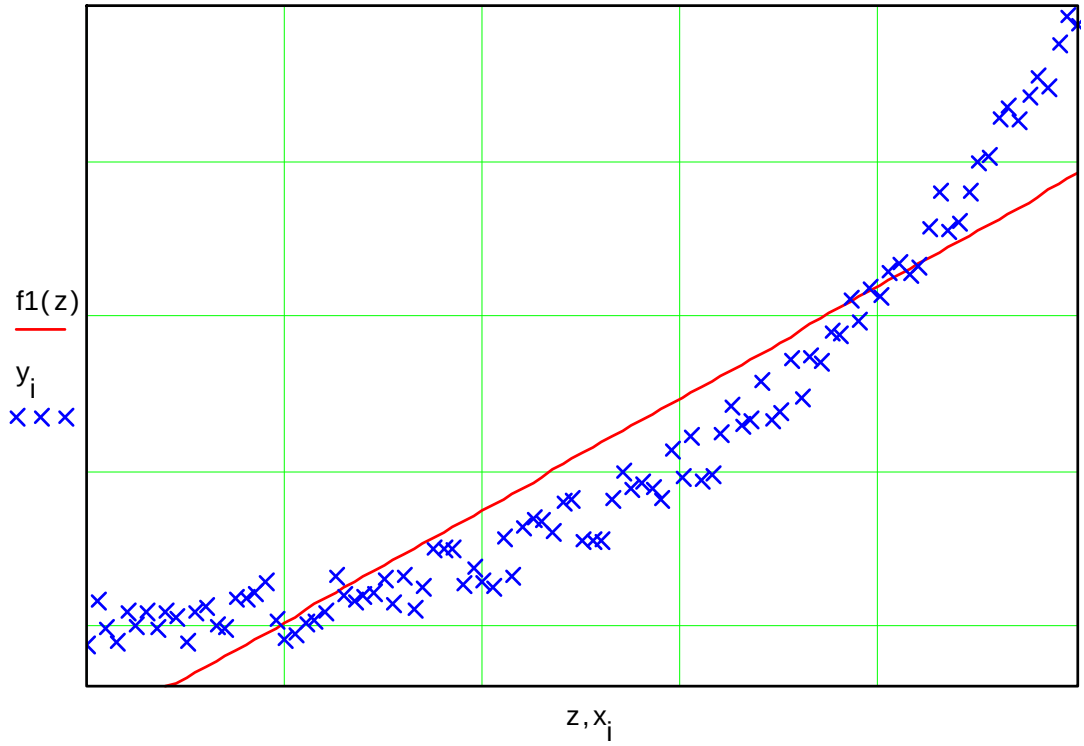
$$\text{line}(x, y) = \begin{pmatrix} -3.539 \\ 1.81 \end{pmatrix}$$

Вычислим стандартное отклонение:

$$S1 := \sqrt{\frac{1}{(n-2)} \cdot \sum_i \left(f1(x_i) - y_i\right)^2}$$

$S1 = 2.093$

$z := 0, 0.1 .. 10$



3.3.2. Аппроксимация полиномами

Теперь попытаемся подобрать полиномы второй и третьей степени, в качестве аппроксимирующей функции. Для этих целей служат встроенные функции ***regress*** и уже знакомая нам функция ***interp***. (Очевидно, что если в качестве аппроксимирующей функции брать полином степени на единицу меньше числа точек, то задача сведется к задаче глобальной интерполяции и полученный полином будет точно проходить через все заданные узлы.)

Вводим степени полиномов:

k2 := 2

k3 := 3

Функция ***regress*** является вспомогательной, она подготавливает данные, необходимые для работы функции ***interp***. Вектор ***vs*** содержит в том числе и коэффициенты полинома

vs2 := regress (x , y , k2)

vs3 := regress (x , y , k3)

Функция ***interp*** возвращает значение полинома в точке ***z***. Определив новые функции `f2, f3`, мы получили возможность находить значе-

ние полинома в любой заданной точке:

f2 (z) := interp (vs2 , x , y , z)

f3 (z) := interp (vs3 , x , y , z)

coeffs2 := submatrix (vs2 , 3 , length (vs2) − 1 , 0 , 0)

coeffs3 := submatrix (vs3 , 3 , length (vs3) − 1 , 0 , 0)

Коэффициенты:

$(coeffs2)^T = (0.701 \quad -0.76 \quad 0.257)$

$(coeffs3)^T = (-0.122 \quad 0.253 \quad 2.377 \cdot 10^{-3} \quad 0.017)$

$$S2 := \sqrt{\left[\frac{1}{(n-2)} \cdot \sum_i \left(f2(x_i) - y_i\right)^2\right]} \qquad S2 = 0.671$$

$$S3 := \sqrt{\left[\frac{1}{(n-2)} \cdot \sum_i \left(f3(x_i) - y_i\right)^2\right]} \qquad S3 = 0.581$$

Стандартные отклонения почти не отличают друг от друга, коэффициент при четвертой степени z невелик, поэтому дальнейшее увеличение степени полинома нецелесообразно и достаточно ограничиться только второй степенью.

К сожалению, функция ***regress*** имеется далеко не во всех версиях ***MathCAD***. Однако, провести полиномиальную регрессию можно и без использования этой функции. Для этого нужно определить коэффициенты нормальной системы и решить полученную систему уравнений, например, матричным методом.

Теперь попытаемся аппроксимировать экспериментальные данные полиномами степени m и m1, не прибегая к помощи встроенной функции ***regress***:

m := 2

t := 0 .. m

j := 0 .. m

m1 := 3

t1 := 0 .. m1

j1 := 0 .. m1

Вычисляем элементы матрицы коэффициентов нормальной системы

$$p_{t,j} := \sum_i \left(x_i\right)^{t+j}$$

$$p1_{t1,j1} := \sum_i \left(x_i\right)^{t1+j1}$$

и столбец свободных членов

$$b_j := \sum_i y_i \cdot \left(x_i\right)^j$$

$$b1_{j1} := \sum_i y_i \cdot \left(x_i\right)^{j1}$$

Находим коэффициенты полинома, решая систему матричным методом:

$$a := p^{-1} \cdot b \qquad a1 := p1^{-1} \cdot b1$$

Определяем аппроксимирующие функции

$$f2(z) := \sum_t a_t \cdot z^t$$

$$f3(z) := \sum_{t1} a1_{t1} \cdot z^{t1}$$

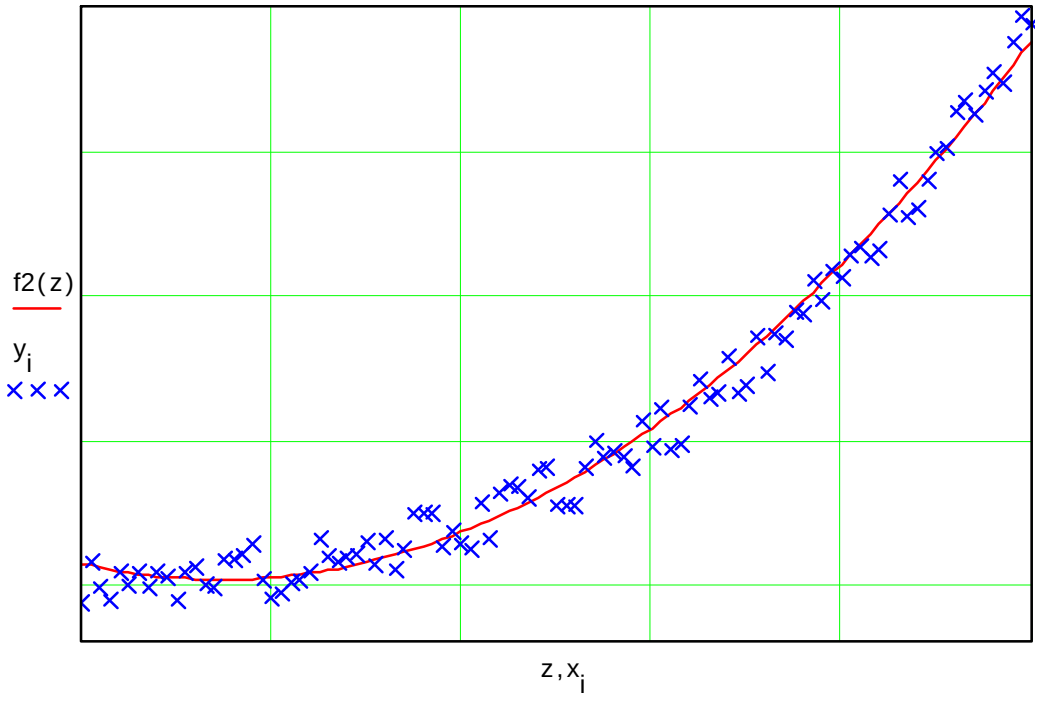
Коэффициенты полиномов следующие:

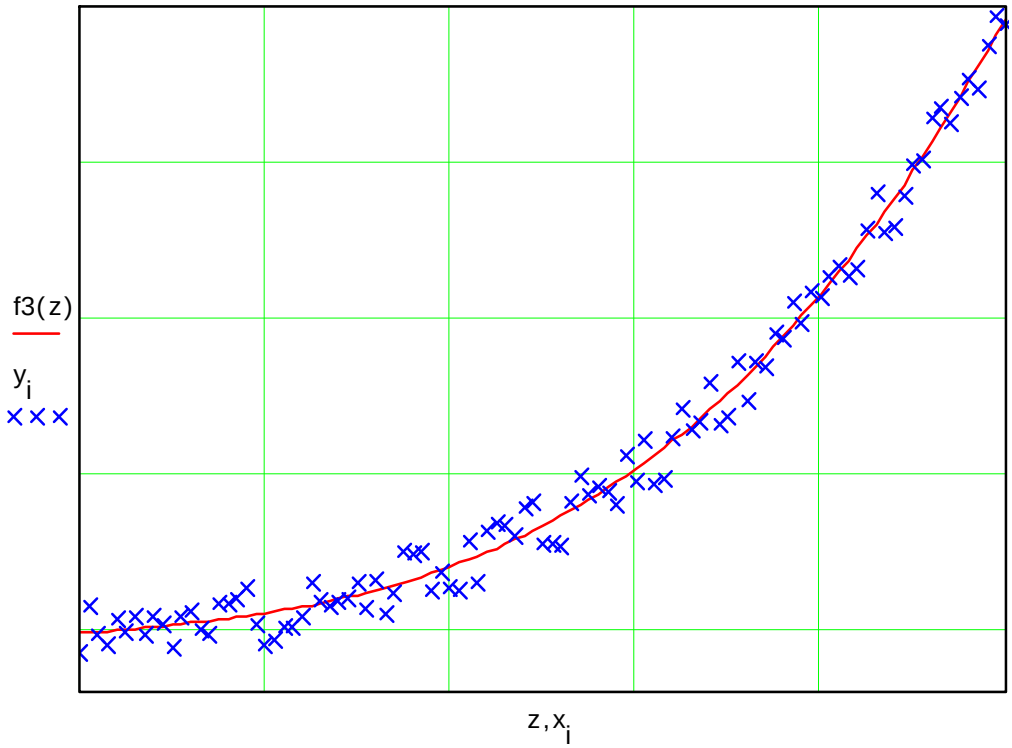
$$a = \begin{pmatrix} 0.701 \\ -0.76 \\ 0.257 \end{pmatrix} \qquad a1 = \begin{bmatrix} -0.122 \\ 0.253 \\ 2.377 \cdot 10^{-3} \\ 0.017 \end{bmatrix}$$

Вычислим стандартное отклонение

$$S2 := \sqrt{\left[\frac{1}{(n-2)} \cdot \sum_i \left(f2\left(x_i\right) - y_i\right)^2\right]} \qquad S2 = 0.671$$

$$S3 := \sqrt{\left[\frac{1}{(n-2)} \cdot \sum_i \left(f3\left(x_i\right) - y_i\right)^2\right]} \qquad S3 = 0.581$$


f2(z)
y_i
z, x_i


f3(z)
y_i
z, x_i

3.3.3. Аппроксимация линейной комбинацией функций

MathCAD предоставляет пользователям встроенную функцию ***linfit*** для аппроксимации данных по методу наименьших квадратов линейной комбинацией произвольных функций.

Функция ***linfit*** имеет три аргумента:

- вектор ***x*** – x – координаты заданных точек;
- вектор ***y*** – y – координаты заданных точек;
- функция ***F*** – содержит набор функций, который будет использоваться для построения линейной комбинации.

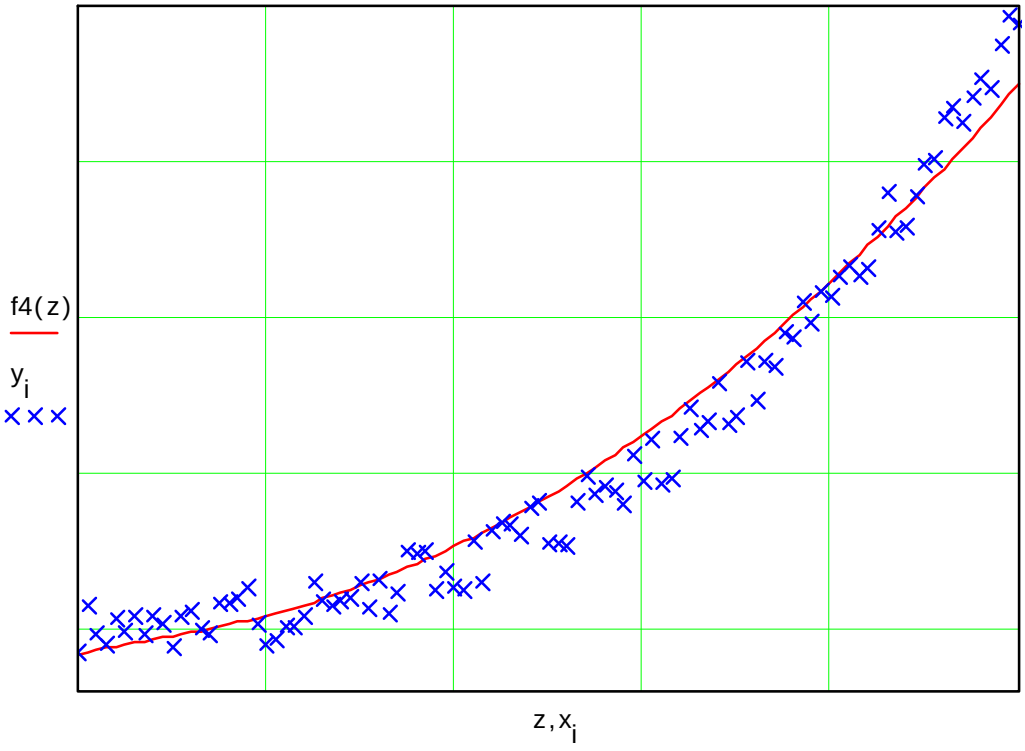
Задаем функцию F (аппроксимирующая функция) ищется в виде $a\frac{1}{x+1} + b\cdot x^2$):

$$F(x) := \begin{pmatrix} \frac{1}{x+1} \\ x^2 \end{pmatrix} \qquad S := \text{linfit}(x, y, F) \qquad S = \begin{pmatrix} -0.802 \\ 0.176 \end{pmatrix}$$

Определяем аппроксимирующую функцию: $f4(x) := F(x) \cdot S$

Вычисляем дисперсию:

$$S4 := \sqrt{\left[\frac{1}{(n-2)} \cdot \sum_i \left(f4(x_i) - y_i\right)^2\right]} \qquad S4 = 0.975$$



3.3.4. Аппроксимация функцией произвольного вида

Теперь построим аппроксимирующую функцию дробно-рационального типа $f(x)=\frac{ax^2}{b+x}$. Для этого воспользуемся функцией ***genfit***. Функция имеет следующие параметры:

- ***x***, ***y*** – векторы, содержащие координаты заданных точек;
- ***F*** – функция, задающая искомую функциональную *n*-параметрическую зависимость и частные производные этой зависимости по параметрам;
- ***v*** – вектор, задающий начальные приближения для поиска параметров;

$$F(z,u):=\begin{bmatrix}\frac{u_0\cdot z^2}{u_1+z}\\ \frac{z^2}{u_1+z}\\ -\frac{u_0\cdot z^2}{(u_1+z)^2}\end{bmatrix}$$

$$v:=\begin{pmatrix}-1\\-15\end{pmatrix}$$

$$S:=\text{genfit}(x,y,v,F)\qquad S=\begin{pmatrix}-2.146\\-20.85\end{pmatrix}$$

Поскольку нулевой элемент функции *F* содержит искомую функцию, определяем функцию следующим образом:

$$f5(z):=F(z,S)_0$$

Вычисляем среднее квадратичное отклонение

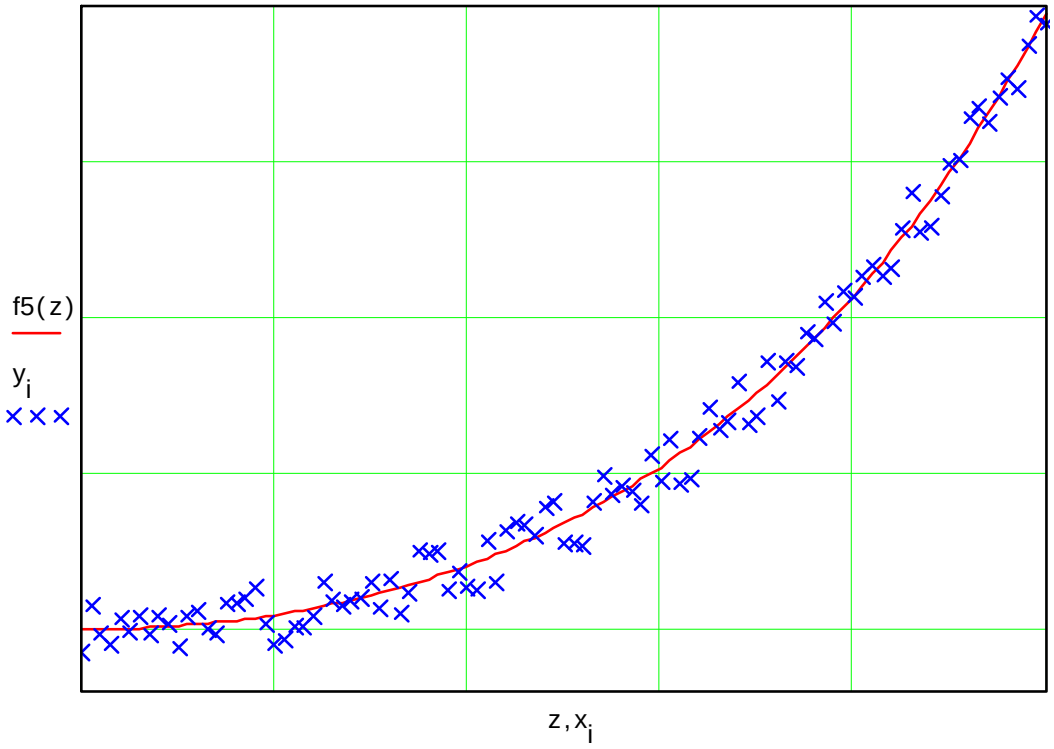
$$S5:=\sqrt{\left[\frac{1}{(n-2)}\cdot\sum_i\left(f5(x_i)-y_i\right)^2\right]}\qquad S5=0.581$$

Функция ***genfit*** имеется не во всех реализациях ***MathCAD***. Возможно, однако, решить задачу, проведя линеаризацию.

Заданная функциональная зависимость может быть линеаризована введением переменных $z=\frac{1}{y}$ и $t=\frac{1}{x}$.

Тогда

$$z = \frac{1}{a} + b\frac{t}{a}.$$



Определим матрицы коэффициентов нормальной системы [8]:

$$e := \begin{bmatrix} \sum_i \left[(x_i)^3 \cdot y_i \right] \\ \sum_i \left[x_i \cdot (y_i)^2 \right] \end{bmatrix} \qquad o := \begin{bmatrix} \sum_i (x_i)^4 & -\left[\sum_i (x_i)^2 \cdot y_i \right] \\ \sum_i \left[(x_i)^2 \cdot y_i \right] & -\left[\sum_i (y_i)^2 \right] \end{bmatrix}$$

Находим коэффициенты функции, решая систему матричным методом:

$$d := o^{-1} \cdot e \qquad d = \begin{pmatrix} -1.218 \\ -15.517 \end{pmatrix}$$

Определяем функцию:

$$f5(z) := \frac{\left(z^2 \cdot d_0\right)}{\left(d_1 + z\right)}$$

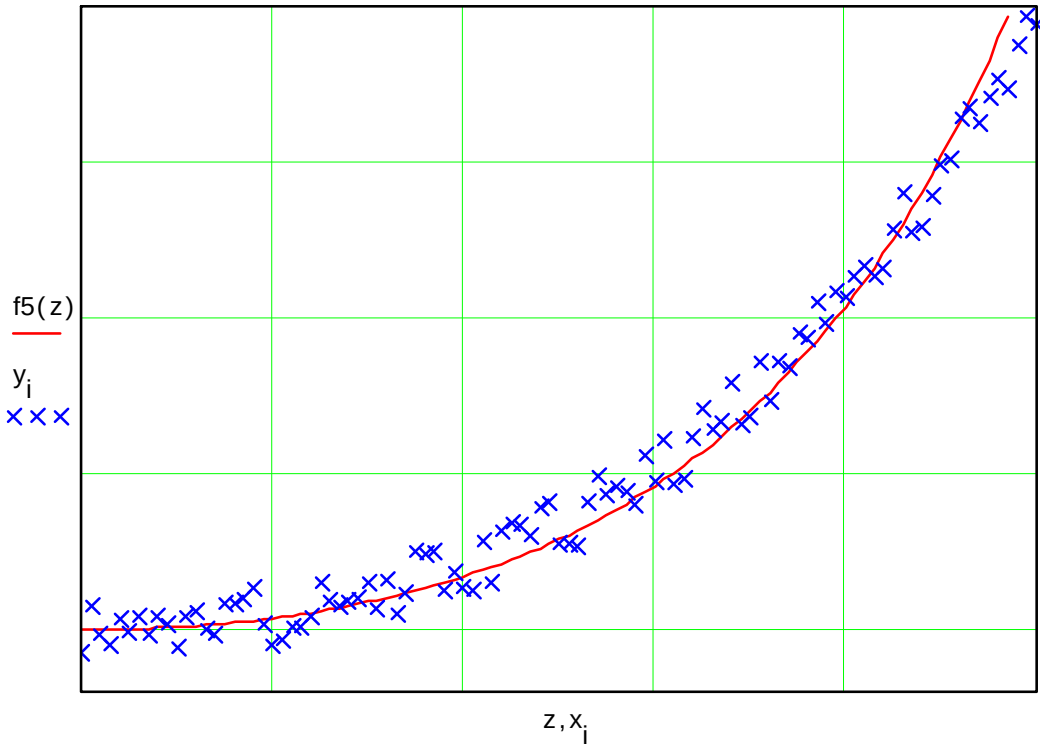
Вычисляем стандартное отклонение

$$S5 := \sqrt{\left[\frac{1}{(n-2)}\cdot\sum_i \left(f5\left(x_i\right)-y_i\right)^2\right]} \qquad S5 = 0.827$$

***Обратите внимание*!** Мы получили другие коэффициенты! Вспомните, задача на нахождение минимума нелинейной функции, особенно нескольких переменных, может иметь несколько решений.

Стандартное отклонение больше, чем в случае аппроксимации полиномами, поэтому следует остановить свой выбор на аппроксимации полиномом.

Представим результаты аппроксимации на графиках.



В тех случаях, когда функциональная зависимость оказывается достаточно сложной, может оказаться, что самый простой способ нахождения коэффициентов – минимизация функционала Ф «в лоб».

4. ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ

Рассмотрим интегрирование функций с использованием формул Гаусса, обладающих наивысшей алгебраической точностью. Простейшие формулы трапеций, средних, Симпсона приведены в виде краткой сводки (без вывода).

4.1. Квадратурная формула Гаусса

Пусть функция задана на стандартном интервале $[-1,\ 1]$. Задача состоит в том, чтобы подобрать точки $t_1,\ t_2,\ldots,t_n$ и коэффициенты $A_1,\ A_2,\ldots,A_n$ так, чтобы квадратурная формула

$$\int_{-1}^{1} f(t)\,dt = \sum_{i=1}^{n} A_i\, f(t_i) \qquad (4.1)$$

была точной для всех полиномов наивысшей возможной степени.

Ввиду того, что имеется $2n$ параметров A_i и t_i $(i=1,\ 2,\ldots,n)$, а полином степени $2n-1$ определяется $2n$ коэффициентами, эта наивысшая степень в общем случае $N=2n-1$.

Запишем полином в виде $f(t)=\sum_{k=0}^{2n-1} a_k\, t^k$ и подставим в (4.1):

$$\int_{-1}^{1} \sum_{k=0}^{2n-1} a_k\, t^k dt = \sum_{i=1}^{n} A_i \sum_{k=0}^{2n-1} a_k\, t_i^k;$$

$$\sum_{k=0}^{2n-1} a_k \int_{-1}^{1} t^k dt = \sum_{k=0}^{2n-1} a_k \sum_{i=1}^{n} A_i\, t_i^k\ .$$

Приравнивая выражения при одинаковых коэффициентах a_k, получим уравнения

$$\int_{-1}^{1} t^k\, dt = \sum_{i=1}^{n} A_i\, t_i^k\ ;\ k=0,\ 1,\ 2,\ldots,\ 2n-1;$$

$$\int_{-1}^{1} t^k dt = \frac{1-(-1)^{k+1}}{k+1} = \begin{cases} \dfrac{2}{k+1}, & k=2\cdot j, \\ 0, & k=2\cdot j+1, \end{cases} \qquad j=0,\ 1,\ldots$$

Итак, A_i и t_i находят из системы $2n$ уравнений:

$$\begin{aligned}
&\sum_{i=1}^{n} A_i = 2;\\
&\sum_{i=1}^{n} A_i\, t_i = 0;\\
&\sum_{i=1}^{n} A_i\, t_i^2 = \frac{2}{3};\\
&\cdots\cdots\\
&\sum_{i=1}^{n} A_i\, t_i^{2n-1} = 0.
\end{aligned} \qquad (4.2)$$

Система (4.2) нелинейная и ее решение найти довольно трудно. Рассмотрим еще один прием нахождения A_i и t_i. Свойства полиномов Лежандра вида

$$P_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}[(x^2-1)^n], \quad n = 0,\ 1,\ 2, \ldots$$

таковы:

1) $P_n(1) = 1, \quad P_n(-1) = (-1)^n, \qquad n = 0,\ 1,\ 2, \ldots;$

2) $\int_{-1}^{1} P_n(x) P_m(x)\, dx = \delta_{mn} N_m, \quad N_m = \frac{2}{2n+1};$

3) полином Лежандра $P_n(x)$ имеет n различных и действительных корней, расположенных на интервале $[-1,\ 1]$.

Составим по узлам интегрирования многочлен n-й степени

$$w_n(x) = \prod_{k=1}^{n} (x - x_k).$$

Функция $f(x) = w_n(x)\, P_m(x)$ при $m \le n-1$ есть многочлен степени не выше $2n-1$. Значит для этой функции формула Гаусса справедлива:

$$\int_{-1}^{1} w_n(x)\, P_m(x)\, dx = \sum_{i=1}^{n} A_i\, w_n(x_1)\, P_m(x_i) = 0, \qquad (4.3)$$

так как $w_n(x) = 0$.

Разложим $w_n(x)$ в ряд по ортогональным многочленам Лежандра:

$$w_n(x) = \sum_{k=0}^{n} b_k P_k(x);$$

$$\int_{-1}^{1} w_n(x) P_m(x)\,dx = \sum_{k=0}^{n} b_k P_k(x) P_m(x) = b_m N_m = 0;$$

$$m \le n-1,$$

т.е. все коэффициенты $b_m = 0$ при $m \le n-1$. Значит $w_n(x)$ с точностью до численного множителя совпадает с $P_n(x)$. Таким образом, узлами формулы Гаусса являются нули многочлена Лежандра степени n.

Зная t_i, из линейной теперь системы первых n (4.2) легко найти коэффициенты $A_i\ (i = 1, 2, \ldots, n)$. Определитель этой системы есть определитель Вандермонда.

Формулу $\int_{-1}^{1} f(t)dt = \sum_{i=1}^{n} A_i f(t_1)$, в которой t_i – нули полинома Лежандра $P_n(t)$, а A_i определяют из (4.3), называют квадратурной формулой Гаусса.

Пример. *Вывести квадратурную формулу Гаусса для случая трех ординат* $(n = 3)$.

Полином Лежандра третьей степени

$$P_3(t) = \frac{1}{2}(5t^3 - 3t).$$

Корни:

$$t_1 = -\sqrt{\frac{3}{5}}, \quad t_2 = 0, \quad t_3 = \sqrt{\frac{3}{5}}.$$

Из (4.2) имеем

$$A_1 + A_2 + A_3 = 2;$$

$$-\sqrt{\frac{3}{5}}A_1 + \sqrt{\frac{3}{5}}A_3 = 0;$$

$$\frac{3}{5}A_1 + \frac{3}{5}A_3 = \frac{2}{3}.$$

Отсюда

$$A_1 = A_3 = \frac{5}{9}, \quad A_2 = \frac{8}{9}.$$

Тогда

$$\int_{-1}^{1} f(t)dt = \frac{1}{9}\left[5f\left(-\sqrt{3/5}\right)+8f(0)+5f\left(\sqrt{3/5}\right)\right].$$

Рассмотрим теперь применение квадратурной формулы Гаусса для вычисления интеграла с не единичными пределами $\int_a^b f(x)dx$:

$$x = \frac{b+a}{2} + \frac{b-a}{2}t.$$

Получим

$$\int_a^b f(x)\,dx = \frac{b-a}{2}\int_{-1}^{1} f\left(\frac{b+a}{2}+\frac{b-a}{2}t\right)dt;$$

$$\int_a^b f(x)\,dx = \frac{b-a}{2}\sum_{i=1}^{n} A_i f(x_i),$$

где

$$x_i = \frac{b+a}{2} + \frac{b-a}{2}t_i,\ i=1,\ 2,\ldots,n;$$

t_i – нули полинома Лежандра $P_n(t)$, т.е. $P_n(t_i)=0$.

Остаточный член формулы Гаусса с узлами выражается формулой

$$R_n = \frac{(b-a)^{2n+1}(n!)^4}{(2n+1)\ \ [(2n)!]^3} f^{(2n)}(\xi).$$

Отсюда следует:

$$R_2 = \frac{1}{135}\left(\frac{b-a}{2}\right)^5 f^{(4)}(\xi);$$

$$R_3 = \frac{1}{15750}\left(\frac{b-a}{2}\right)^7 f^{(6)}(\xi);$$

$$R_4 = \frac{1}{3472875}\left(\frac{b-a}{2}\right)^9 f^{(8)}(\xi)$$

и т.д.

4.2. Простейшие квадратурные формулы

Приведем сводку ряда простейших формул, используемых в практике численного интегрирования.

Формула трапеций:

$$\int_a^b f(x)dx \approx h\left(\frac{1}{2}f_0 + f_1 + \ldots + f_{N-1} + \frac{1}{2}f_N\right);$$

$$R \approx -\frac{1}{12}h^2\int_a^b f''(x)dx = O(h^2); \qquad (4.4)$$

$$h = x_i - x_{i-1} = \text{const},$$

где R – погрешность формулы;

f_i – значения интегрируемой функции в узлах.

Формула средних:

$$\int_a^b f(x)\,dx \approx h\sum_{i=1}^{N} f_{i-\frac{1}{2}};$$

$$R \approx \frac{1}{24}h^2\int_a^b f''(x)\,dx = O(h^2).$$

Формула Симпсона:

$$\int_a^b f(x)\,dx \approx \frac{h}{3}\sum_{i=0}^{\frac{N}{2}-1}(f_{2i} + 4f_{2i+1} + f_{2i+2}); \qquad (4.5)$$

$$R \approx -\frac{1}{180}h^4\int_a^b f^{(4)}(x)\,dx = O(h^4).$$

4.3. Метод Монте-Карло

Хотя ***MathCAD*** позволяет вычислять кратные интегралы непосредственно, однако в большинстве случаев при кратности интегралов трех и более применение метода Монте-Карло предпочтительнее. Дело в том, при одинаковой точности метод Монте-Карло дает существенный выигрыш во времени (в десятки и сотни раз), особенно при большой кратности интегралов. Идея метода состоит в том, что интеграл заменяется величиной $F_{cp} \cdot V$, где V – объем области интегрирования, F_{cp} – среднее значение подынтегральной функции, вычисленное по нескольким случайно выбранным точкам.

Определим подынтегральную функцию

$$f(x,y,z) := 125 - x^2 - y^2 - z^2$$

и вычислим интеграл обычным способом (***обратите внимание на время счета!***):

$$\int_0^1 \int_0^1 \int_0^1 f(x,y,z)\,dz\,dy\,dx = 124$$

А теперь вычислим тот же интеграл методом Монте-Карло:

$N := 3000$ $\quad i := 0 .. N - 1$

$x_i := rnd(1)$ $\quad y_i := rnd(1)$ $\quad z_i := rnd(1)$

$v_i := f(x_i, y_i, z_i)$ $\quad mean(v) = 123.$

Поскольку в нашем случае объем области интегрирования равен 1, полученное среднее значение совпадает со значением интеграла. При относительной погрешности в 0,001 % время вычисления интеграла по методу Монте-Карло существенно меньше.

Интеграл можно вычислить и другим способом. Заключим область интегрирования внутрь прямоугольной области, «набросаем» внутрь полученной области N случайных точек. Тогда интеграл найдем из соотношения $I=\frac{n}{N}\cdot V$, где N – общее число точек, n – число точек, лежащих внутри области интегрирования, V – объем области, включающей область интегрирования.

Максимальное значение подынтегральной функции в области интегрирования не превосходит 125, следовательно, мы может заключить всю область интегрирования внутрь четырехмерного цилиндроида высотой 125 и объемом V = 125. Сгенерируем N четверок случайных чисел и подсчитаем, сколько из них лежит под поверхностью f(x,y,z):

$t_i := rnd(125)$ $\quad S := \sum_i if(t_i < f(x_i, y_i, z_i), 1, 0)$

$I := 125 \cdot \frac{S}{N}$ $\quad I = 124.1$

5. РЕШЕНИЕ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

5.1. Обыкновенные дифференциальные уравнения

Пусть необходимо найти решение уравнения

$$y' = f(x, y) \tag{0.8}$$

с начальным условием $y(x_0) = y_0$. Такая задача называется ***задачей Коши***. Разложим искомую функцию $y(x)$ в ряд вблизи точки x_0 и ограничимся первыми двумя членами разложения:

$$y(x) = y(x_0) + y'(x)(x - x_0) + \ldots .$$

Учтя уравнение (0.8) и обозначив $x - x_0 = h$, получаем

$$y(x) = y(x_0) + f(x_0, \quad y_0)\Delta x.$$

Эту формулу можно применять многократно, находя значения функции во все новых и новых точках:

$$y_{i+1} = y_i + f(x_i, \quad y_i)h . \tag{0.9}$$

Такой метод решения обыкновенных дифференциальных уравнений называется ***методом Эйлера***. Геометрически метод Эйлера означает, что на каждом шаге мы аппроксимируем решение (интегральную кривую) отрезком касательной, проведенной к графику решения в начале интервала. Точность метода невелика и имеет порядок h. Говорят, что метод Эйлера – метод первого порядка, то есть его точность растет линейно с уменьшением шага h.

Существуют различные модификации метода Эйлера, позволяющие увеличить его точность. Все они основаны на том, что производную, вычисленную в начале интервала, заменяют на среднее значение производной на данном интервале. Среднее значение производной можно получить (конечно же, только приближенно) различными способами. Можно, например, оценить значение производной в середине интервала

$$[x_i, \quad x_{i+1}]$$

и использовать его для аппроксимации решения на всем интервале

$$x_{i+\frac{1}{2}} = x_i + \frac{h}{2};$$

$$y_{i+\frac{1}{2}} = y_i + \frac{h}{2} f(x_i, \quad y_i);$$

$$y_{i+1} = y_i + h\, f\left(x_{i+\frac{1}{2}}, \quad y_{i+\frac{1}{2}} \right).$$

Можно также оценить среднее значение производной на интервале

$$\tilde{y}_{i+1} = y_i + h\, f(x_i, \quad y_i);$$

$$y_{i+1} = y_i + h\, \frac{f(x_i, \quad y_i) + f(x_{i+1}, \quad \tilde{y}_{i+1})}{2}.$$

Такие модификации метода Эйлера имеет уже точность второго порядка.

Оценку значения производной можно улучшить, увеличивая число вспомогательных шагов. На практике наиболее распространенным методом решения обыкновенных дифференциальных уравнений ***является метод Рунге – Кутты*** четвертого порядка. Для оценки значения производной в этом методе используется четыре вспомогательных шага. Формулы метода Рунге – Кутты следующие:

$$k_1^i = h\, f(x_i, \quad y_i);$$

$$k_2^i = h\, f\left(x_i + \frac{h}{2}, \quad y_i + \frac{k_1^i}{2} \right);$$

$$k_3^i = h\, f\left(x_i + \frac{h}{2}, \quad y_i + \frac{k_2^i}{2} \right);$$

$$k_4^i = h\, f(x_i + h, \quad y_i + k_3^i);$$

$$\Delta y_i = \frac{1}{2}(k_1^i + 2k_2^i + 2k_3^i + k_4^i);$$

$$y_{i+1} = y_i + \Delta y_i.$$

Перечисленные методы можно применять и для решения систем дифференциальных уравнений. Поскольку многие дифференциальные уравнения высших порядков могут быть сведены заменой переменных к системе дифференциальных уравнений первого порядка, рассмотренные методы могут быть использованы и для решения дифференциальных уравнений порядка выше первого.

Еще один тип задач, часто встречающихся на практике, – краевые задачи. Пусть имеется дифференциальное уравнение второго порядка $y'' = f(x, y, y')$. Решение уравнения требуется найти на интервале $[0, 1]$, причем известно, что $y(0) = y_0$, $y(1) = y_1$. Понятно, что произвольный интервал $[a, b]$ заменой переменных $t = \frac{x-a}{b-a}$ может быть сведен к единичному. Для решения краевой задачи обычно применяют ***метод стрельб***. Пусть $y'(0) = k$, где k – некоторый параметр. Для некоторого пробного значения k может быть решена задача Коши, например, методом Рунге – Кутты. Полученное решение будет зависеть от значения параметра $y = y(x, k)$. Мы хотим найти такое значение параметра, чтобы выполнялось условие $y(1, k) = y_1$. Фактически мы свели исходную задачу к задаче решения трансцендентного уравнения с таблично заданной функцией. Если найдены такие значения параметра k_1 и k_2, что $y(1, k_1) > y_1$ и $y(1, k_1) > y_1$, то дальнейшее уточнение значения параметра можно проводить методом деления отрезка пополам.

5.1.1. Метод Эйлера для дифференциальных уравнений первого порядка

Решим задачу Коши для дифференциального уравнения первого порядка $y' = f(x, y)$ методом Эйлера.

Пусть правая часть уравнения равна $f(x, y) \equiv x \cdot y$

Зададим границы изменения x: $x_{min} \equiv 0$ $x_{max} \equiv 1$

Зададим число точек и величину шага: $x_{min} \equiv 0$ $x_{max} \equiv 1$

Зададим начальные условия: $y_0 \equiv 1$ $x_0 \equiv x_{min}$

Вычислим x и y по формулам Эйлера $j \equiv 1 .. n$:

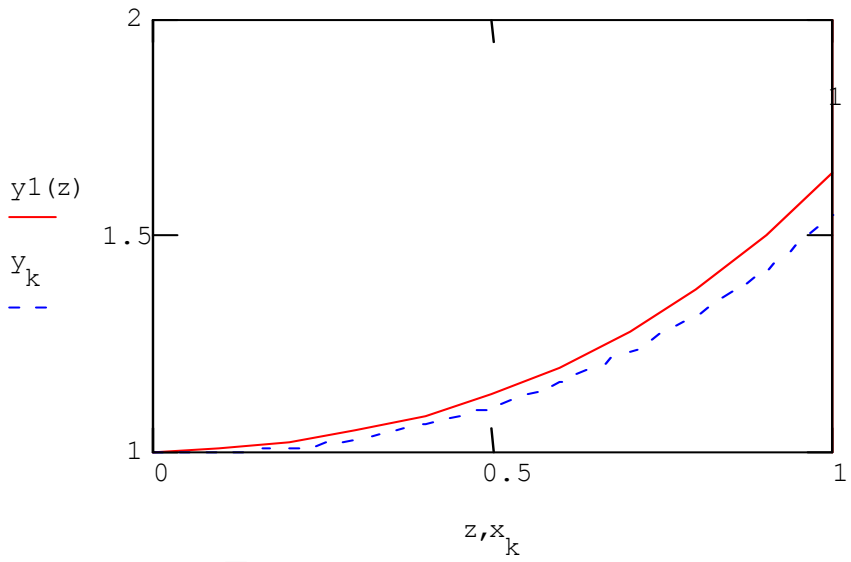
$x_j \equiv x_{min} + j \cdot h$

$y_j \equiv y_{j-1} + f(x_{j-1}, y_{j-1}) \cdot h$

Представим результат графически и сравним его с аналитическим решением:

$y1(x) := \exp\left(\frac{x^2}{2}\right)$

$z := 0, 0.1 .. 1$ $k \equiv 0 .. n$



Точное решение

Метод Эйлера

Точное аналитическое решение и решение, полученное численно, отличаются в точке $x = 1$ на $y1(1) - y_n = 0.102$.

То есть относительная ошибка составляет $\frac{y1(1) - y_n}{y1(1)} = 6.163 \cdot \%$.

5.1.2. Решение систем дифференциальных уравнений

Для решения дифференциальных уравнений ***MathCAD*** имеет ряд встроенных функций, в частности, функцию ***rkfixed***, реализующую метод Рунге – Кутты четвертого порядка с фиксированным шагом. Фактически эта функция предназначена для решения систем дифференциальных уравнений первого порядка:

$$y_1' = f_1(x, \ y_1, \ y_2, \ldots, y_n);$$
$$y_2' = f_2(x, \ y_1, \ y_2, \ldots, y_n);$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$y_n' = f_n(x, \ y_1, \ y_2, \ldots, y_n).$$

Функция ***rkfixed*(*y*, *x*1, *x*2, *npoints*, *D*)** возвращает матрицу. Первый столбец этой матрицы содержит точки, в которых получено решение, а остальные столбцы – решения и его первые $n-1$ производные.

Аргументы функции:

- ***y*** – вектор начальных значений (***n*** элементов);
- ***x1*** и ***x2*** – границы интервала, на котором ищется решение дифференциального уравнения;
- ***Npoints*** – число точек внутри интервала (***x1***, ***x2***), в которых ищется решение. Функция ***rkfixed*** возвращает матрицу, состоящую из **1+*npoints*** строк;
- ***D*** – вектор, состоящий из n элементов, который содержит первые производные искомой функции.

В качестве примера рассмотрим решение системы Вольтерра – Лотки. Эта система описывает динамику численности хищников и жертв на замкнутом ареале и является одной из базовых моделей экологии:

$$\frac{dN_1}{dt} = N_1(\varepsilon_1 - \gamma_2 N_2);$$
$$\frac{dN_2}{dt} = N_2(\varepsilon_2 - \gamma_1 N_1).$$

Для решения систем дифференциальных уравнений используются функция ***rkfixed***.

Внимание*!** В этом примере установлено значение ***ORIGIN*=1**, то есть нумерация элементов массива начинается с единицы, а не с нуля, как это принято в ***MathCAD по умолчанию.

Пусть в начальный момент времени число хищников $N_1 = 5$ и число жертв $N_2 = 10$. Задаем:

- вектор начальных значений $N := \begin{pmatrix} 5 \\ 10 \end{pmatrix}$;
- параметры системы $\varepsilon := \begin{pmatrix} 0.1 \\ 0.3 \end{pmatrix}$ $\gamma := \begin{pmatrix} 0.03 \\ 0.04 \end{pmatrix}$;
- интервал времени и количество точек, в которых будет вычислено решение $t_{max} := 200$ $npoints := 400$;
- вектор правых частей системы (поскольку исходная система не зависит явно от времени ***t***, функция ***D*** так же не зависит от времени явно, хотя и содержит его в числе своих аргументов)

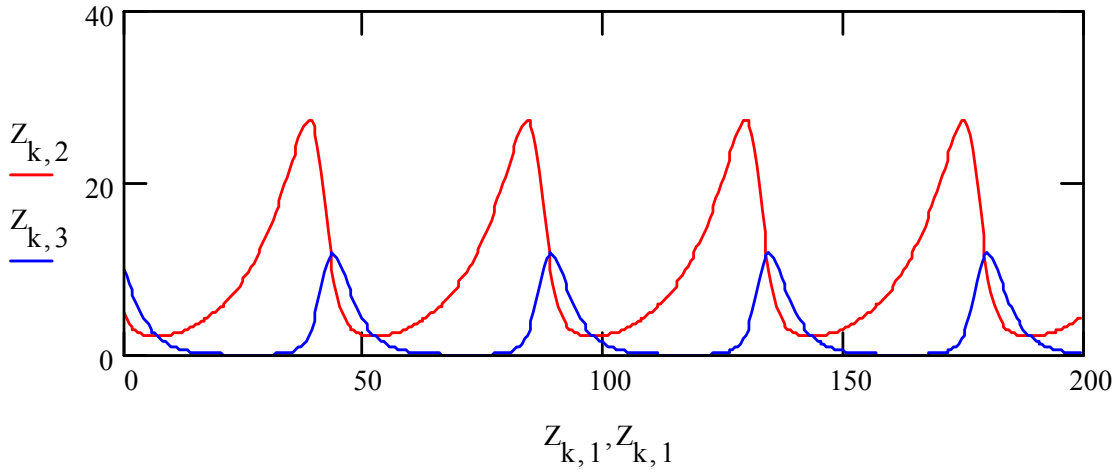
$$D(t, N) := \begin{bmatrix} N_1 \cdot (\varepsilon_1 - \gamma_2 \cdot N_2) \\ -N_2 \cdot (\varepsilon_2 - \gamma_1 \cdot N_1) \end{bmatrix}.$$

Решаем систему с помощью встроенной функции

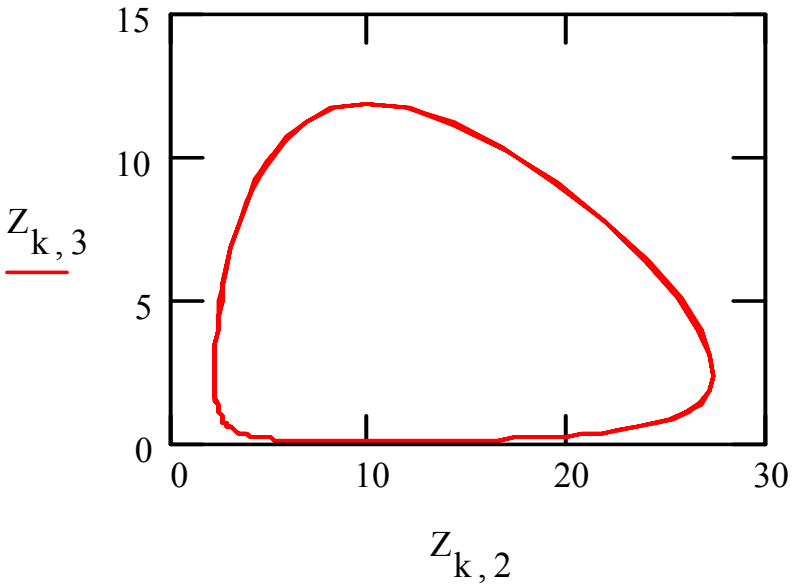
$Z := \text{rkfixed}(N, 0, t_{max}, npoints, D)$

$k := 1..npoints$

Представим на графике результаты расчета – зависимость численности популяций от времени



и зависимость числа жертв от числа хищников:



Можно использовать обозначения $(Z^{\langle i \rangle})_k$ или $Z_{k,i}$ – это одно и то же. Поскольку дифференциальное уравнение порядка выше первого может быть преобразовано к системе дифференциальных уравнений первого порядка, функция ***rkfixed*** может быть использована и для решения дифференциальных уравнений.

5.1.3. Решение дифференциальных уравнений методом Рунге – Кутты

Решим еще раз задачу Коши для дифференциального уравнения первого порядка $y' = xy$ методом Рунге – Кутты.

Зададим границы изменения x: $x_{min} := 0$ $x_{max} := 1$.

Зададим число точек внутри интервала: $n := 10$.

Зададим начальные условия $y_0 := 1$.

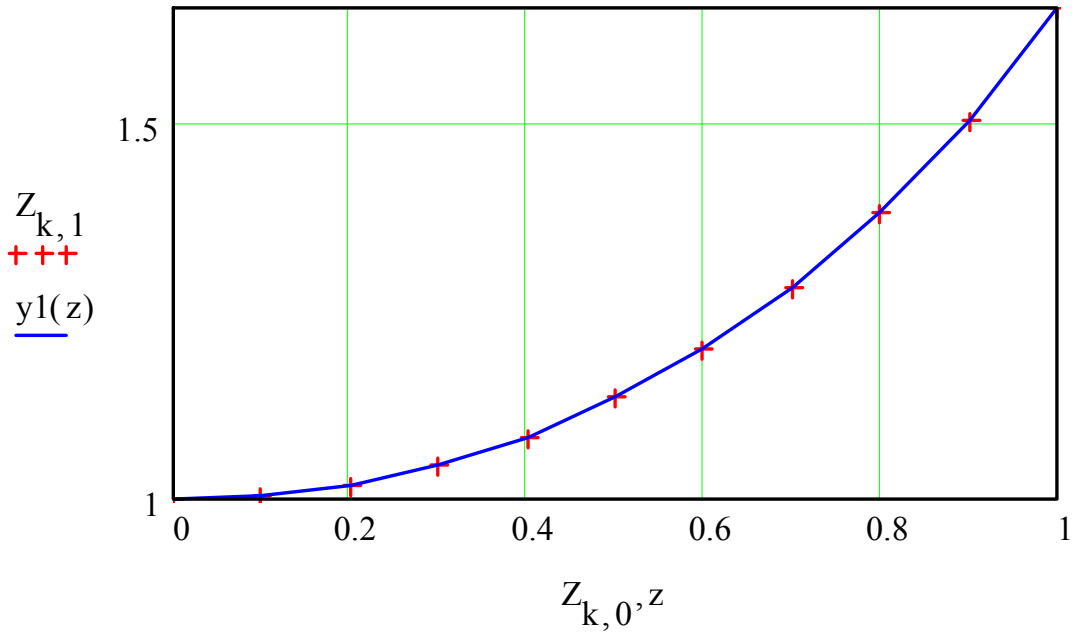
Обратите внимание на обозначения! Поскольку мы решаем только одно дифференциальное уравнение первого порядка, а не систему дифференциальных уравнений, матрица ***y*** содержит только один элемент, однако запись ***y* = 1** была бы неправильной. Необходимо явно указывать на то, что величина ***y*** – матрица, то есть писать индекс.

Определим теперь матрицу производных. Эта матрица тоже состоит только из одного элемента. Этот элемент с точностью до обозначений совпадает с правой частью исходного дифференциального уравнения: $D(x, y) := y_0 \cdot x$.

Решаем дифференциальное уравнение:

$Z := \text{rkfixed}(y, x_{min}, x_{max}, n, D)$

$k := 0 .. n$ $\quad y1(x) := \exp\left(\frac{x^2}{2}\right)$ $\quad z := 0, 0.1 .. 1$



Точное аналитическое решение и решение, полученное численно, отличаются в точке $x=1$ на $y1(1) - \left(Z^{<1>}\right)_n = 2.636 \cdot 10^{-7}$.

Относительная ошибка составляет

$$\frac{\left[y1(1) - \left(Z^{<1>}\right)_n\right]}{y1(1)} = 1.599 \cdot 10^{-5} \cdot \%.$$

5.1.4. Решение дифференциальных уравнений второго порядка

В качестве примера решим задачу о гармоническом осцилляторе, для которого известно аналитическое решение и легко может быть оценена точность вычислений. Дифференциальное уравнение второго порядка

$$y'' + 2\beta y' + \omega^2 y = 0$$

преобразуем к системе из двух дифференциальных уравнений первого порядка

$$y' = x;$$
$$x' = -2\beta x - \omega^2 y.$$

Пусть декремент затухания $\beta := 0.0$. Пусть циклическая частота $\omega := 1 \cdot$ Зададим начальные условия: $y := \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

Значение ***y0*** соответствует начальной координате, а ***y1*** – начальной скорости. Зададим теперь матрицу ***D***. С учетом того, что искомая величина соответствует нулевому элементу массива ***y***, ее первая производная – первому, а вторая – второму, имеем

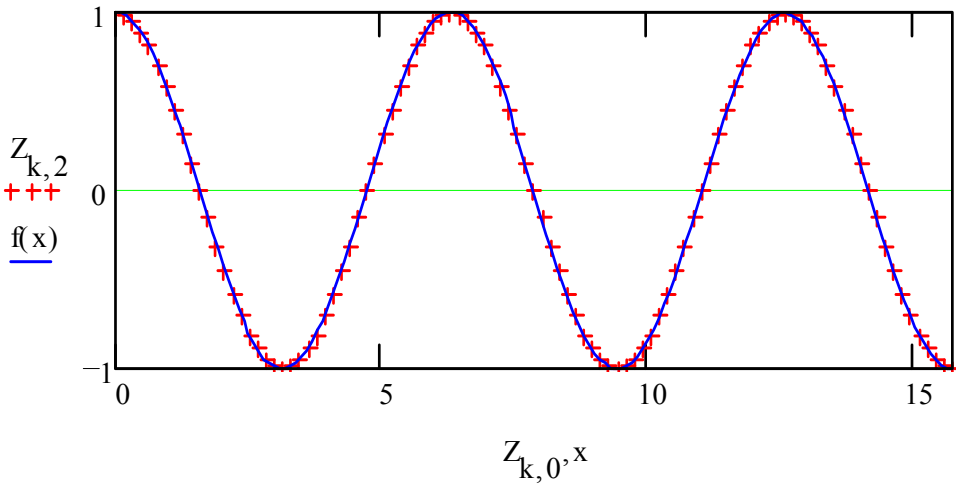
$$D(t, y) := \begin{pmatrix} y_1 \\ -2 \cdot \beta \cdot y_1 - \omega^2 \cdot y_0 \end{pmatrix};$$

$Z := \text{rkfixed}(y, 0, 5 \cdot \pi, 100, D)$.

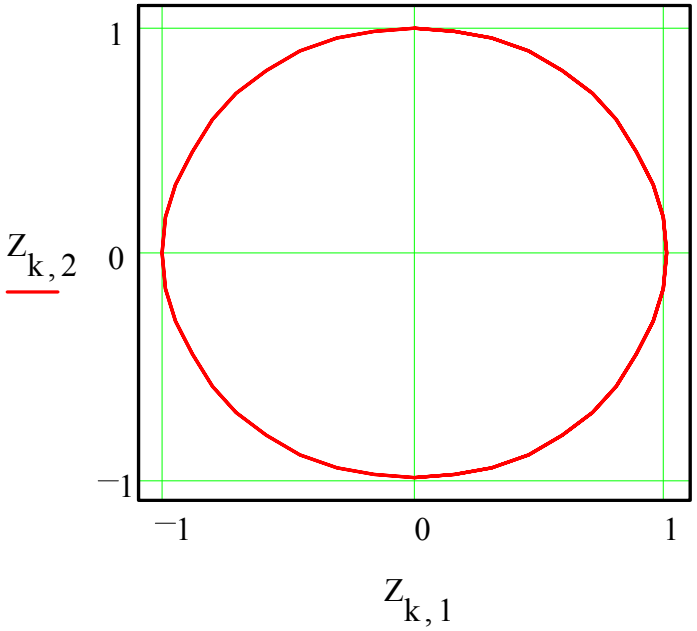
Представим результаты расчета на графике и сравним их с аналитическим решением $f(x) := y_1 \cdot \exp(-\beta \cdot x) \cdot \cos(\omega \cdot x)$

k01 $x := 0, 0.1 .. 5 \cdot \pi$

k01



Для контроля точности вычислений нарисуем фазовую траекторию (зависимость смещения от скорости). Для гармонического осциллятора фазовая траектория должна иметь вид эллипса.



Примечание: ***MathCAD*** имеет еще две функции для решения задачи Коши. Это функции ***Rkadapt*** и ***Bulstoer***. Эти функции имеют те же самые аргументы и возвращают решения в такой же форме, что и функция ***rkfixed***. Первая из этих функций использует метод Рунге – Кутты с переменным шагом, что позволяет повысить точность вычислений и сократить их объем, если искомое решение имеет области, где ее значения меняются быстро, и области плавного изменения. Функция ***Rkadapt*** будет варьировать величину шага в зависимости от скорости

изменения решения.

Функция ***Bulstoer*** реализует иной численный метод – метод Булирша – Штёра. Ее следует применять, если известно, что решение является гладкой функцией.

5.1.5. Решение краевой задачи

Пусть требуется найти решение дифференциального уравнения $y'' + 2\beta y + \omega_0^2 y = 0$ при условиях $y(0) = 1$ и $y(5\pi) = 0$.

При значениях параметров $\beta := 0.1$; $\omega_0 := 1$.

Для решения краевой задачи имеется встроенная функция ***sbval***, реализующая метод стрельб и позволяющая свести краевую задачу к задаче Коши.

Функция ***sbval*** имеет следующие параметры:

- ***v*** – вектор, содержащий начальные приближения для недостающих начальных условий;
- x_{min}, x_{max} – границы интервала, на котором ищется решение;
- ***D(x, y)*** – вектор-функция, содержащий правые части системы дифференциальных уравнений первого порядка, эквивалентной исходному уравнению, размер вектора ***n*** совпадает со степенью старшей производной дифференциального уравнения;
- ***load(xmin,v)*** – вектор-функция, элементы которой соответствуют ***n*** значениям функций на левой границе интервала. Часть этих значений известна, а для части заданы начальные приближения в векторе ***v***. Их уточненные значения будут найдены в процессе вычисления;
- ***score(xmax,y)*** – вектор-функция, имеющая то же число элементов, что и ***v***. Каждое значение является разностью между начальными значениями в конечной точке интервала и соответствующей оценки для решения. Этот вектор показывает, насколько близко найденное решение к истинному.

Наша задача сводится к системе двух дифференциальных уравнений первого порядка:

$$y' = z;$$
$$z' = -2\beta z - \omega_0^2 y.$$

Поэтому функция D имеет вид

$$D(t, y) := \begin{pmatrix} y_1 \\ -2 \cdot \beta \cdot y_1 - \omega_0^2 \cdot y_0 \end{pmatrix}$$

Задаем граничные условия:

$x_{min} := 0; \qquad x_{max} := 5 \cdot \pi .$

Задача Коши для дифференциального уравнения второго порядка содержит два начальных условия. Нам известно только одно. Начальное приближение для недостающего значения задаем в векторе ***v***, который в нашем случае состоит только из одного элемента. Несмотря на это, индекс 0 должен быть обязательно указан, чтобы подчеркнуть векторный характер этой величины: $v_0 := 0.1$.

На левой границе интервала нам известно значение $y\ (y(0)=1)$ и задано начальное приближение для $y'\ (y'=0)$. Это значение записано в v_0. Задаем вектор-функцию ***load***. Ее нулевой элемент – начальное значение для y_0, первый – для y_1:

$$load\left(x_{min}, v\right) := \begin{pmatrix} 1 \\ v_0 \end{pmatrix} \qquad score\left(x_{max}, y\right) := y_0 - 0$$

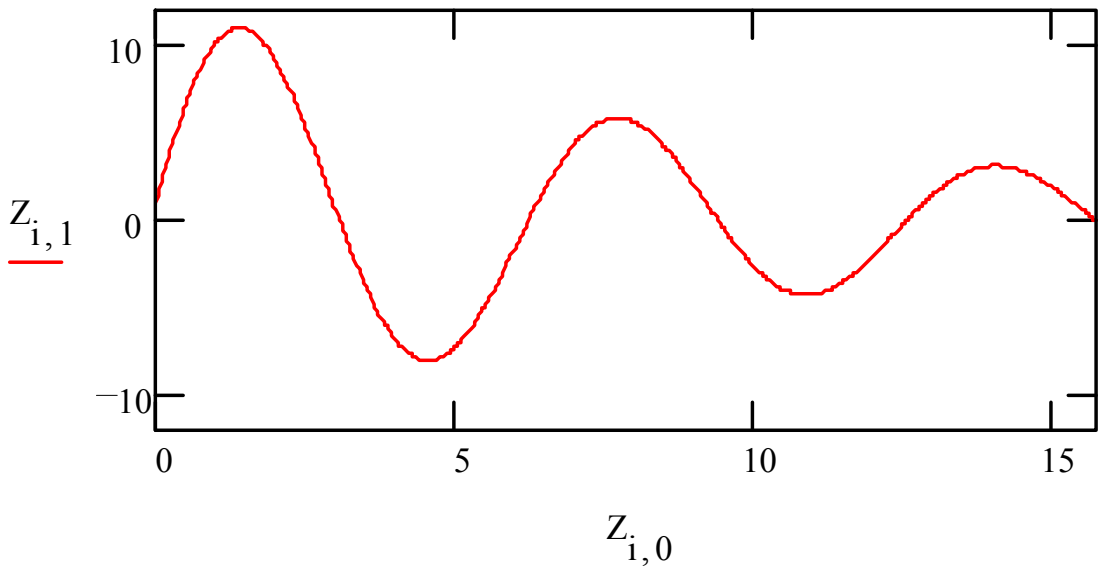
$$S := sbval\left(v, x_{min}, x_{max}, D, load, score\right)$$

$$S = 12.511$$

Теперь, когда нам стало известно недостающее начальное условие в задаче Коши, можно воспользоваться, например, функцией ***rkfixed***:

$$n := 500 \qquad i := 0..n \qquad y := \begin{pmatrix} 1 \\ S_0 \end{pmatrix}$$

$$Z := rkfixed\left(y, x_{min}, x_{max}, 500, D\right)$$



5.1.6. Решение обыкновенных дифференциальных уравнений в MathCAD

MathCAD предлагает новый способ для решения обыкновенных дифференциальных уравнений, разрешенных относительно старшей производной. Для этих целей служит уже известный нам блок ***given*** совместно с функцией ***odesolve***. Дифференциальное уравнение совместно с начальными или граничными условиями записывается в блоке ***given***. Производные можно обозначать как штрихами (***Ctrl+F7***), так и с помощью знака производной $\frac{d}{dx}$. Приведем пример использования функции для решения задачи Коши:

$\beta := 0.1$

Given

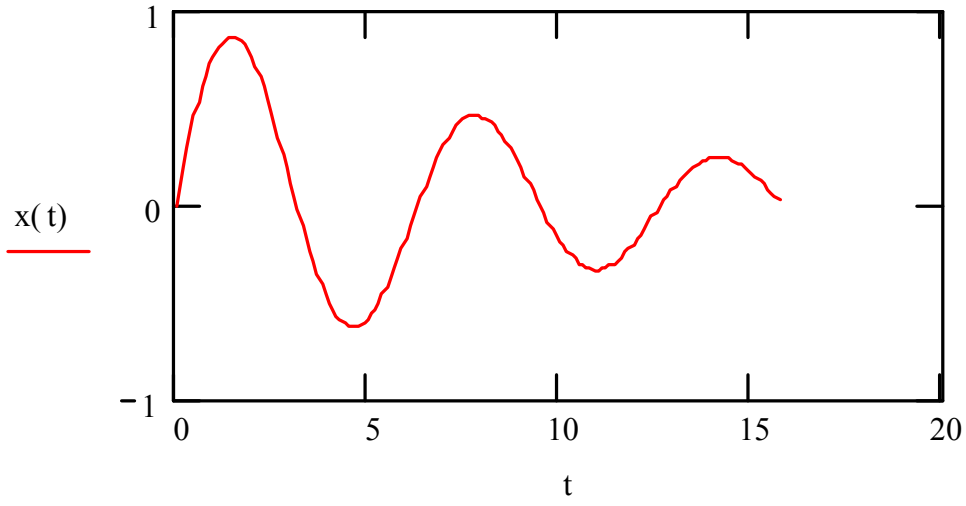
$x''(t) + 2 \cdot \beta \cdot x'(t) + x(t) = 0$

$x(0) = 0$

$x'(0) = 1$

$x := \text{odesolve}(t, 5 \cdot \pi, 0.1)$

$t := 0, 0.1 .. 5 \cdot \pi$



Обратите внимание! У искомой функции явно указан аргумент, знак производной стоит перед скобкой.

Функция ***odesolve*** имеет три аргумента. Первый аргумент – независимая переменная, вторая – граница интервала, на котором ищется решение, последний аргумент – шаг, с которым ищется решение. Последний аргумент может быть опущен.

Следующий пример демонстрирует решение краевой задачи. Использован другой способ записи производных, используется ***odesolve*** функция с двумя аргументами:

Given

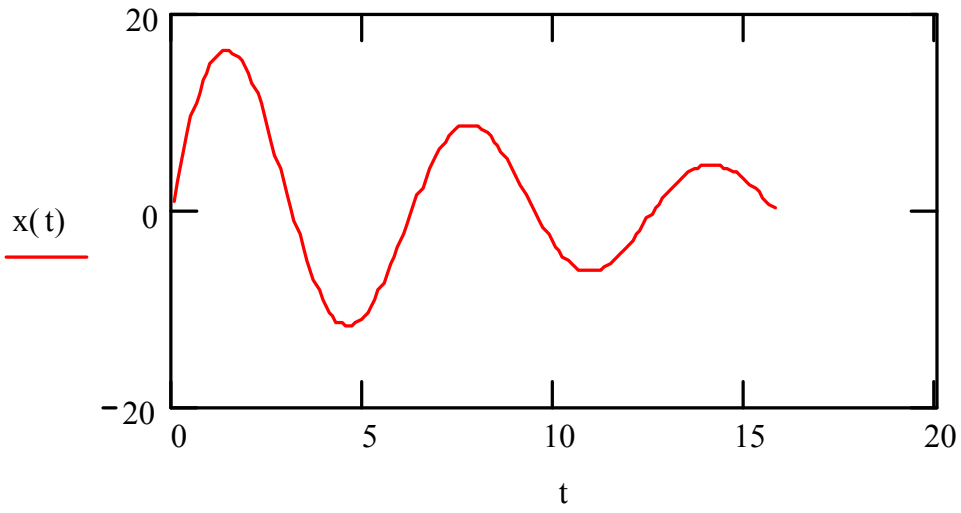
$$\frac{d^2}{dt^2}x(t)+2\cdot\beta\cdot\frac{d}{dt}x(t)+x(t)=0$$

$$x(0)=1$$

$$x(5\cdot\pi)=0.1$$

$$x:=\text{odesolve}\ (t,5\cdot\pi)$$

$$t:=0,0.1..5\cdot\pi$$



5.2. Решение уравнений в частных производных

Одним из методов решения дифференциальных уравнений в частных производных является ***метод сеток***. Идея метода заключается в следующем. Для простоты, ограничимся случаем только функции двух переменных, и будем полагать, что решение уравнения ищется на квадратной области единичного размера. Разобьем область сеткой. Шаг сетки по оси x и по оси y, вообще говоря, может быть разный. По определению частная производная

$$\frac{\partial u(x,\ y)}{\partial x}=\lim_{\Delta x\to 0}\frac{u(x+\Delta x,\ y)-u(x)}{\Delta x}\approx\frac{u(x+\Delta x,\ y)-u(x)}{\Delta x}.$$

Если рассматривать функцию только в узлах сетки, то частную производную можно записать в форме

$$\frac{\partial u(x,\ y)}{\partial x} \approx \frac{u_{i+1,j} - u_{i,j}}{h},$$

где узел $(i,\ j)$ соответствует точке $(x,\ y)$. Полученное выражение называется ***правой конечной разностью***. Название связано с тем, что для вычисления производной в точке используются значение функции в этой точке и точке, лежащей правее. Очевидно, что сходное выражение можно было бы получить, используя точку, лежащую слева:

$$\frac{\partial u(x,\ y)}{\partial x} \approx \frac{u_{i,j} - u_{i-1,j}}{h}.$$

Такое выражение называется левой конечной разностью. Можно получить центральную конечную разность, найдя среднее этих выражений.

Теперь получим выражения для вторых производных

$$\frac{\partial^2 u(x,\ y)}{\partial x^2} \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}.$$

В данном случае для нахождения производной мы использовали симметричные точки. Однако, очевидно, можно было бы использовать точки с несимметричным расположением.

5.2.1. Уравнения гиперболического типа

В качестве примера рассмотрим решение волнового уравнения (уравнения гиперболического типа)

$$\frac{\partial^2 U}{\partial\, x^2} = \frac{1}{v^2}\frac{\partial^2 U}{\partial\, t^2}.$$

Уравнение будем решать методом сеток. Запишем уравнение в конечных разностях

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} = \frac{1}{v^2}\frac{u_{i,j+1} - 2u_{i,j} + u_{i,j}}{\tau^2}.$$

Это уравнение позволяет выразить значение функции `u` в момент времени $j+1$ через значения в предыдущие моменты времени:

$$u_{i,j+1} = v^2\left(\frac{\tau}{h}\right)^2 (u_{i+1,j} - 2u_{i,j} + u_{i-1,j}) + 2u_{i,j} - u_{i,j-1}.$$

Такая разностная схема называется явной, так как искомая величина получается в явном виде. Она устойчива, если $\tau \le h/v$.

Зададим начальные условия: смещение струны U в начальный и последующий моменты времени описывается синусоидальной функцией:

$n := 20$ $\quad j := 0 .. n$ $\quad i := 0 .. 100$

$U_{i,0} := \sin\left(\pi \cdot \frac{i}{50}\right)$ $\quad U_{i,1} := U_{i,0}$

(Совпадение смещений при j=0 и j=1 соответствует нулевой начальной скорости.)

Зададим граничные условия: на концах струны смещение равно нулю в любой момент времени: $U_{0,j} := 0$ $\quad U_{100,j} := 0$

Будем полагать:

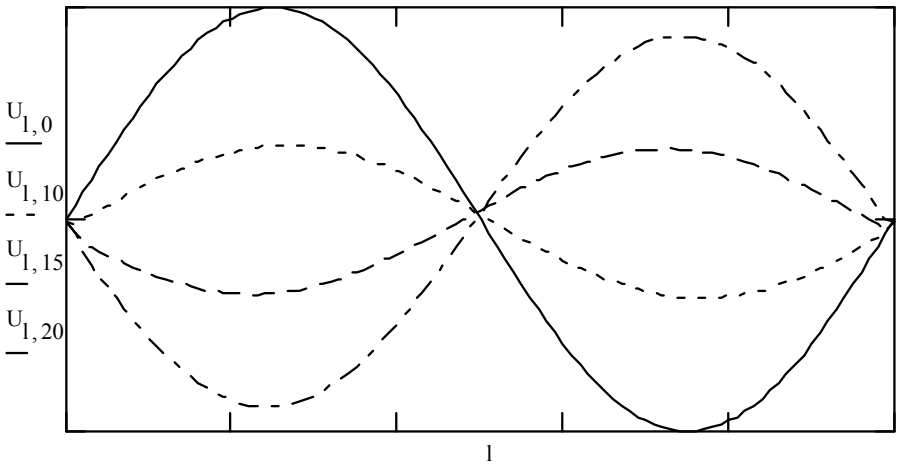
$a := 1$ $\quad k := 0.02$ $\quad i := 1 .. 99$

$j := 1 .. n - 1$ $\quad l := 0 .. 100$

Записываем уравнение в конечных разностях, разрешенное относительно $U_{i,j+1}$:

$$U_{i,j+1} := a^2 \cdot k \cdot \left(U_{i+1,j} - 2 \cdot U_{i,j} + U_{i-1,j}\right) + 2 \cdot U_{i,j} - U_{i,j-1}$$

Представляем результат на графике.



5.2.2. Уравнения параболического типа

Еще один пример использования конечных разностей – уравнение диффузии

$$\frac{\partial u}{\partial t} = D \frac{\partial^2 u}{\partial x^2}.$$

Это уравнение параболического типа. Явная разностная схема для этого уравнения имеет вид

$$u_{i,j+1} = D\frac{\tau}{h^2}(u_{i+1,j} - 2u_{i,j} + u_{i-1,j}) + u_{i,j}. \qquad (0.10)$$

Эта разностная схема устойчива, если $\tau \le \frac{h^2}{2D}$. Для краткости в дальнейшем мы будем обозначать весь множитель, стоящий перед скобкой, как □.

Задаем коэффициент к := .15 и диапазон изменения пространственной и временной координат:

t := 0 .. 29 x := 1 .. 49

Задаем начальные и граничные условия:

$f_{0,x} := 0$ $f_{0,0} := 0$

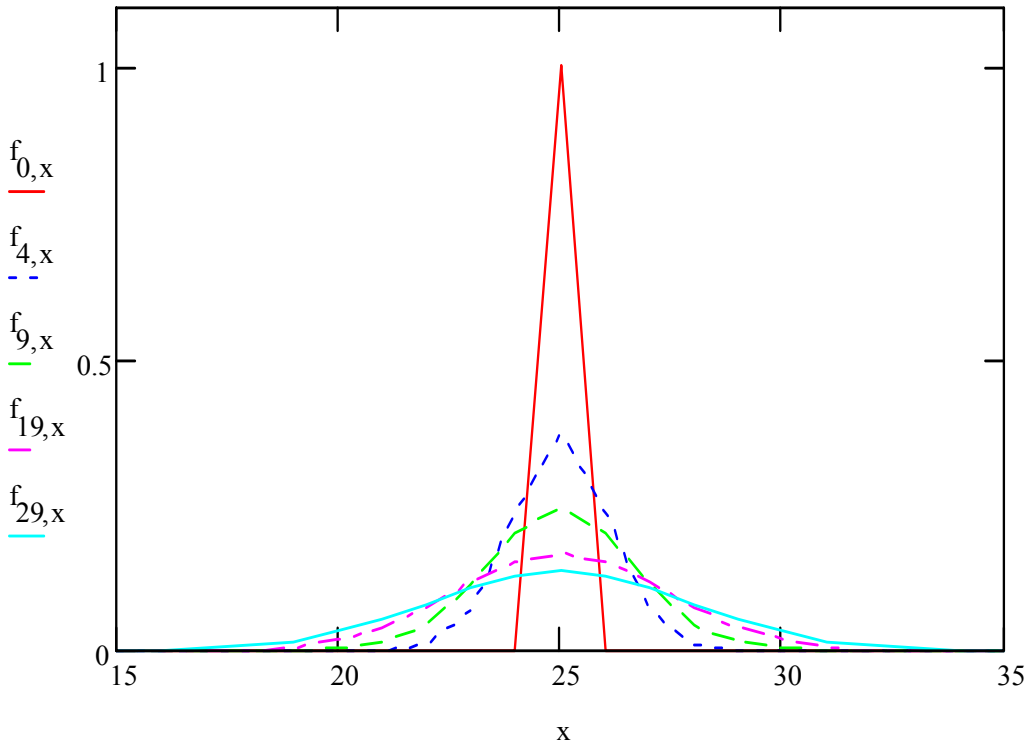
$f_{0,50} := 0$ $f_{0,25} := 1$

Уравнение в конечных разностях имеет вид

$$f_{t+1,x} := f_{t,x} + \kappa \cdot \left(f_{t,x-1} - 2 \cdot f_{t,x} + f_{t,x+1}\right)$$

Представляем результаты на графике (для большей наглядности изображена только центральная часть):

x := 15 .. 35



Основное достоинство явных методов – их простота – зачастую сводится на нет достаточно жесткими ограничениями на величину шага. Явные схемы обычно устойчивы при столь малых шагах по времени, что они становятся непригодными для практических расчетов. Этого существенного недостатка позволяют избежать неявные схемы. Свое название они получили потому, что значения искомой функции на очередном временном шаге не могут быть явно выражены через значения функции на предыдущем шаге.

Рассмотрим применение неявной схемы на примере уравнения теплопроводности

$$\frac{\partial u}{\partial t} = c\frac{\partial^2 u}{\partial x^2}. \qquad (0.11)$$

Запишем неявную разностную схему для этого уравнения:

$$\frac{u_{i,j+1} - u_{i,j}}{\tau} = \frac{c}{h^2}\left(u_{i+1,j+1} - 2u_{i,j+1} + u_{i-1,j+1}\right). \qquad (0.12)$$

Здесь первый индекс соответствует пространственной, а второй – временной координате. В отличие от явной схемы, для вычисления в правой части уравнения используются значения функции на том же самом временном шаге. Вводя обозначение $\mu = \frac{c\tau}{h^2}$, уравнение (0.12) можно переписать в виде

$$(1+2\mu)\,u_{i,j+1} - \mu\,(u_{i+1,j+1} + u_{i-1,j+1}) = u_{i,j} \qquad (0.13)$$

или в матричной форме

$$\begin{pmatrix} 1+2\mu & -\mu & & & & \\ -\mu & 1+2\mu & -\mu & & & \\ & & \cdots & & & \\ & & & 1+2\mu & -\mu & \\ & & & -\mu & 1+2\mu & \end{pmatrix} \begin{pmatrix} u_{1,j+1} \\ u_{2,j+1} \\ \vdots \\ u_{n-1,j+1} \\ u_{n,j+1} \end{pmatrix} = \begin{pmatrix} u_{1,j} + \mu\alpha \\ u_{2,j} \\ \vdots \\ u_{n-1,j} \\ u_{n,j} + \mu\beta \end{pmatrix}, \qquad (0.14)$$

где $u(0,\,t) = \alpha,\; u(t,\,1) = \beta$.

Задаем количество узлов сетки (в данном случае оно одинаково для обеих переменных) n := 30:

i := 0 .. n j := 0 .. n

k := 0 .. n – 1 m := 1 .. n – 1

Задаем значения параметров

$\alpha := 0 \qquad \beta := 1 \qquad \mu := 5$

и начальное распределение температуры в области

$u_{i,0} := \sin\left(\pi \cdot \frac{i}{n}\right) + \frac{i}{n} \cdot$

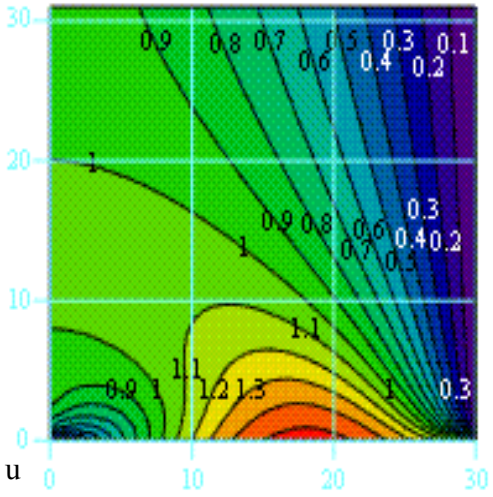
Формируем матрицы уравнения (0.14):

$u_{0,j} := \alpha \qquad u_{n,j} := \beta$

$A_{i,i} := 1 + 2 \cdot \mu \qquad A_{m,m-1} := -\mu \qquad A_{m-1,m} := -\mu$

$\mu\alpha_i := 0 \qquad \mu\alpha_0 := \mu \cdot \alpha \qquad \mu\beta_i := 0 \qquad \mu\beta_0 := \mu \cdot \beta$

Находим решение системы $u^{\langle j+1 \rangle} := A^{-1} \cdot \left(u^{\langle j \rangle} + \mu\alpha + \mu\beta\right)$



5.2.3. Решение уравнений Лапласа и Пуассона

Для решения уравнений Пуассона $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = F(x,\ y)$ и Лапласа (частный случай, когда $F(x,\ y) = 0$) – уравнений эллиптического типа – предназначена функция ***relax*(*a*, *b*, *c*, *d*, *e*, *f*, *u*, *rjac*)**, реализующая метод релаксации. Фактически, эту функцию можно использовать для решения эллиптического уравнения общего вида

$$A\frac{\partial^2 u}{\partial x^2} + 2B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + a\frac{\partial u}{\partial x} + b\frac{\partial u}{\partial y} + cu = F(x,\ y);$$

$$D = AC - B^2 > 0,$$

которое может быть сведено к уравнению в конечных разностях

$$a_{i,j}u_{i+1,j} + b_{i,j}u_{i-1,j} + c_{i,j}u_{i,j+1} + d_{i,j}u_{i,j-1} + e_{i,j}u_{i,j} = f_{i,j}.$$

В частности, для уравнения Пуассона коэффициенты $a_{i,j} = b_{i,j} = c_{i,j} = d_{i,j} = 1, \quad e_{i,j} = -4$.

Идея метода релаксации заключается в следующем. Если нет источников (уравнение Лапласа), то значение функции в данном узле на текущем шаге $k+1$ определяется как среднее значение функции в ближайших узлах на предыдущем шаге k

$$u_{i,j}^{k+1} = \frac{1}{4}(u_{i-1,j}^{k} + u_{i+1,j}^{k} + u_{i,j-1}^{k} + u_{i,j+1}^{k}). \qquad (0.15)$$

При наличии источников разностная схема имеет вид

$$u_{i,j}^{k+1} = \frac{1}{4}(u_{i-1,j}^{k} + u_{i+1,j}^{k} + u_{i,j-1}^{k} + u_{i,j+1}^{k}) - \frac{h^2}{4}f_{i,j}. \qquad (0.16)$$

Метод релаксации сходится достаточно медленно, так как фактически он использует разностную схему (0.10) с максимально возможным для двумерного случая шагом $\tau = \frac{h^2}{4}$.

В методе релаксации необходимо задать начальное приближение, то есть значения функции во всех узлах области, а так же граничные условия.

Функция ***relax*** возвращает квадратную матрицу, в которой:

1) расположение элемента в матрице соответствует его положению внутри квадратной области;

2) это значение приближает решение в этой точке.

Эта функция использует метод релаксации для приближения к решению.

Вы должны использовать функцию ***relax***, если знаете значения искомой функции ***u(x, y)*** на всех четырех сторонах квадратной области.

Аргументы:

a, ***b***, ***c***, ***d***, ***e*** – квадратные матрицы одного и того же размера, содержащие коэффициенты дифференциального уравнения;

f – квадратная матрица, содержащая значения правой части уравнения в каждой точке внутри квадрата;

u – квадратная матрица, содержащая граничные значения функции на краях области, а также начальное приближение решения во внутренних точках области;

rjac – параметр, управляющий сходимостью процесса релаксации.

Он может быть в диапазоне от 0 до 1, но оптимальное значение зависит от деталей задачи:

$n := 2^4$ $\quad i := 0 .. n$ $\quad j := 0 .. n$

Задаем правую часть уравнения Пуассона – два точечных источника:

$M_{i,j} := 0$ $\quad M_{6,8} := 10$ $\quad M_{10,8} := -10$

Задаем значения параметров функции ***relax***:

$a_{i,j} := 1$ $\quad b := a$ $\quad c := a$

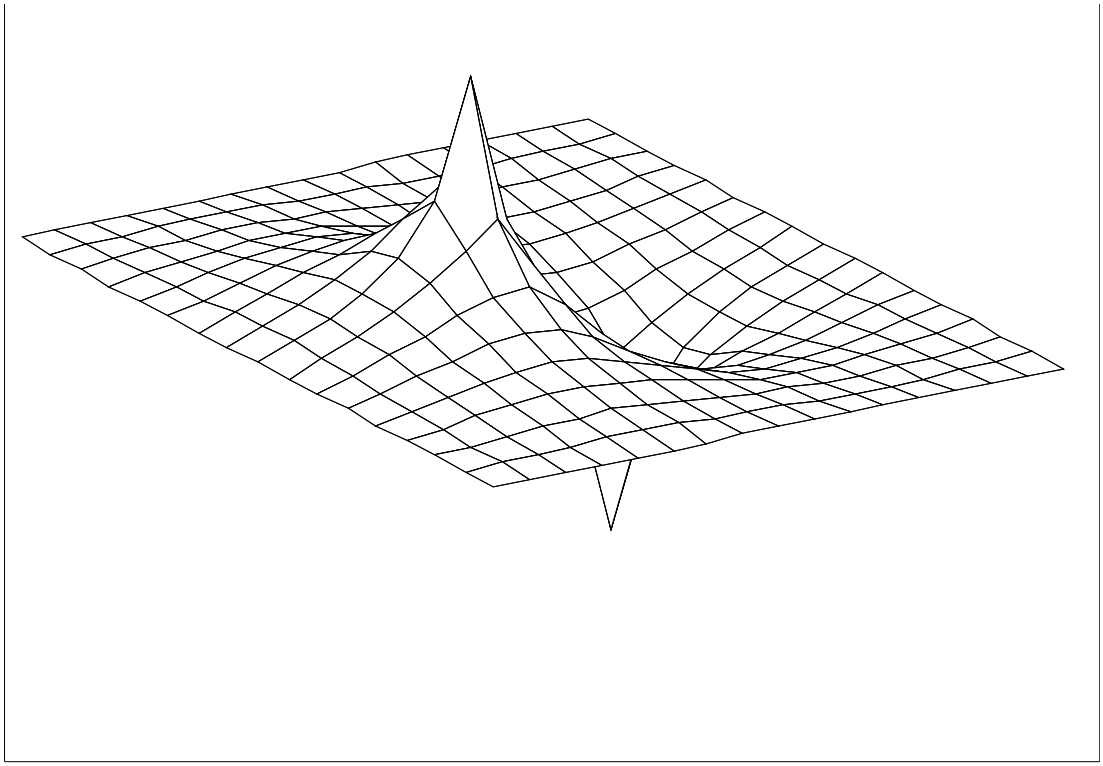
$d := a$ $\quad f := M$ $\quad e := -4 \cdot a$

Задаем граничные условия и начальное приближение – нули во всех внутренних точках области

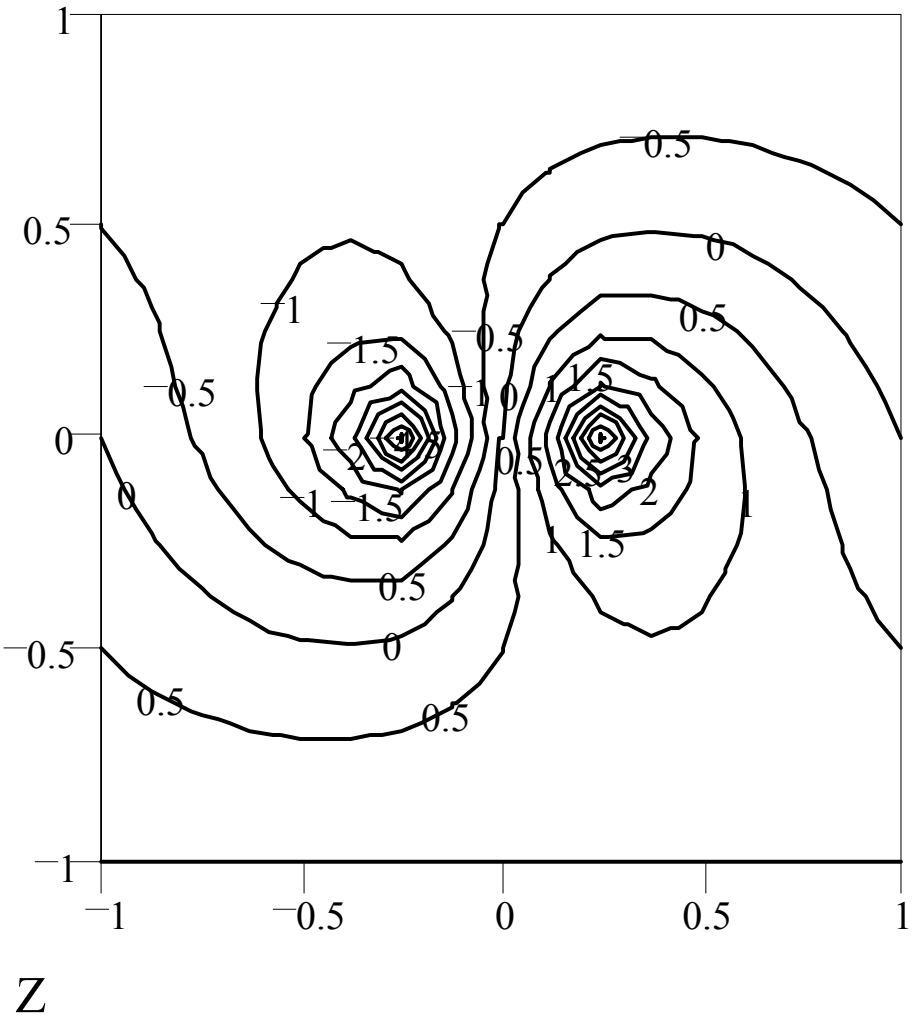
$u_{i,j} := 0$ $\quad u_{i,n} := -1$ $\quad u_{0,j} := 1 - 2 \cdot \frac{j}{n}$

$u_{i,0} := 1$ $\quad u_{n,j} := 1 - 2 \cdot \frac{j}{n}$

Находим решение $Z := relax(a, b, c, d, e, f, u, 0.95)$ и представляем его графически в виде поверхности и линий уровней.



Z



Если граничные условия равны нулю на всех четырех сторонах квадрата, можно использовать функцию ***multigrid***

Z := multigrid (M , 3)

СПИСОК ЛИТЕРАТУРЫ

1. Фридман А. Уравнения с частными производными параболического типа. – М.: Мир, 1968.

2. Тихонов А.Н., Самарский А.А. Уравнения математической физики. – М.: Изд-во МГУ, 1999.

3. Справочник по точным решениям уравнений тепло- и массопереноса / А.Д. Полянин [и др.]. – М.: Факториал, 1998.

4. Цой П.В. Методы решения отдельных задач тепломассопереноса. – М.: Энергия, 1971.

5. Новые направления получения аналитических краевых задач тепломассопереноса и термоупругости для многослойных конструкций / В.А. Кудинов [и др.] // Тепломассообмен. ММФ–2000. – Минск: АНК ИТМО, 2000. – Т. 3. – С. 230–236.

6. Аналитические решения задач взаимосвязанного тепломассопереноса для многослойных конструкций / В.А. Кудинов [и др.] // Тепломассообмен. ММФ–2000. – Минск: АНК ИТМО, 2000. – Т. 3. – С. 402–406.

7. Яненко Н.Н. Метод дробных шагов решения многомерных задач математической физики. – Новосибирск: Наука, 1967.

8. Яненко Н.Н. Об одном разностном методе счета многомерного уравнения теплопроводности // Доклады АН СССР. – 1959. – Т. 125, № 6. – С. 1207–1210.

9. Марчук Г.И. Методы расщепления. – М.: Наука, 1988.

10. Формалев В.Ф., Тюкин О.А. Неявный экономичный метод численного решения задач, содержащих смешанные производные // Математическое моделирование. – 1996. – Т. 8, № 6. – С. 27–33.

11. Формалев В.Ф., Тюкин О.А. Экономичный абсолютно устойчивый метод расщепления с экстраполяцией численного решения задач, содержащих смешанные дифференциальные операторы // Вычислительные технологии: сб. науч. тр. – Новосибирск: Изд-во ВТ СО РАН, 1995. – Т. 4, № 10. – С. 290–299.

12. Рихтмайер Р., Мортон К. Разностные методы решения краевых задач. – М.: Мир, 1972.

13. Doglas J., Gunn J.E. A general formulation of alternating direction methods. Part 1. Parabolic and hyperbolic problems. Numerical Mathematics. – 1964. – V. 6. – P. 428–453.

14. Ковеня В.М., Тарнавский Г.А., Черный С.Г. Применение метода расщепления в задачах аэродинамики. – Новосибирск: Наука, Сибирское отделение, 1990.

15. Ковеня В.М., Яненко Н.Н. Методы расщепления в задачах газовой динамики. – М.: Наука, 1981.
16. Годунов С.К., Забродин А.В., Иванов М.Я. Численное решение многомерных задач газовой динамики. – М.: Наука, 1976.
17. Самарский А.А., Вабищевич П.Н. Численные методы решения задач конвекции-диффузии. – М.: Эдиториал УРСС, 1999.
18. Самарский А.А., Вабищевич П.Н., Матус П.П. Разностные схемы с операторными множителями. – Минск: Изд-во ИММ РАН, ИМ НАНБ, 1998.
19. Самарский А.А., Николаев Е.С. Методы решения сеточных уравнений. – М.: Наука, 1978.
20. Ильин В.П. Методы неполной факторизации для решения алгебраических систем. – М.: Наука, Физматлит, 1995.
21. Булеев Н.И. Пространственная модель турбулентного обмена. – М.: Наука, 1989.
22. Вабищевич П.Н. Численное моделирование. – М.: Наука, 1993.
23. Четверушкин Б.Н. Математическое моделирование задач динамики излучающего газа. – М.: Наука, 1985.
24. Четверушкин Б.Н. Кинетически-согласованные схемы в газовой динамике. – М.: Изд-во МГУ, 1999.
25. Ортега Дж., Рейнболдт В. Итерационные методы решения нелинейных систем уравнений со многими неизвестными. – М.: Мир, 1975.
26. Флетчер К. Вычислительные методы в динамике жидкостей. – М.: Мир, 1991. – Т. 1–2.
27. Ортега Дж., Пул У. Введение в численные методы решения дифференциальных уравнений. – М.: Наука, Физматлит, 1986.
28. Самарский А.А. Теория разностных схем. – М.: Наука, 1989.
29. Самарский А.А. Введение в численные методы. – М.: Наука, 1987.
30. Ильин В.П. Методы конечных разностей и конечных объемов для эллиптических уравнений. – Новосибирск: Изд-во Института математики СО РАН, 2000.
31. Завьялов Ю.С., Квасов Б.С., Мирошниченко В.Л. Методы сплайн-функций. – М.: Наука, 1980.

Учебное издание

ЧИСЛЕННЫЕ МЕТОДЫ

Учебное пособие

Составитель
КРИЦКИЙ Олег Леонидович

В авторской редакции

Верстка *А.А. Цыганкова*

Отпечатано в Издательстве ТПУ в полном соответствии с качеством предоставленного оригинал-макета

Подписано к печати Формат 60×84/16.
Бумага «Снегурочка». Печать Xerox.
Усл. печ. л. 4,01. Уч.-изд. л. 3,63.
Заказ . Тираж экз.

Национальный исследовательский
Томский политехнический университет
Система менеджмента качества
Издательства Томского политехнического университета сертифицирована
NATIONAL QUALITY ASSURANCE по стандарту BS EN ISO 9001:2008

ИЗДАТЕЛЬСТВО ТПУ. 634050, г. Томск, пр. Ленина, 30.
Тел./ факс: 8(3822)56-35-35, www.tpu.ru