Глава 6

Решение дифференциальных уравнений

В этой главе…

- Возможности MathCAD для решения задач с дифференциальными уравнениями
- Решение обычных дифференциальных уравнений и их систем
- Решение уравнений в частных производных
- Резюме

Дифференциальные уравнения (далее просто ДУ) являются основой огромного количества расчетных задач из самых различных областей науки и техники. Данная глава посвящена средствам MathCAD для решения задач, имеющих в своем составе одно или несколько дифференциальных уравнений.

Возможности MathCAD для решения задач с дифференциальными уравнениями

Математический пакет MathCAD 12 обладает средствами для решения как обычных ДУ (неизвестная функция зависит от одного аргумента) и их систем, так и ДУ в частных производных. Конечно, средства MathCAD для решения ДУ имеют некоторые ограничения. Многие классы сложных уравнений не поддерживаются в MathCAD, и даже для поддерживаемых уравнений иногда бывает невозможно достичь приемлемой точности (это касается в первую очередь ДУ в частных производных). Так что для сложных дифференциальных задач автор может посоветовать читателю обратиться к средствам других, более специализированных пакетов, например MATLAB. О подключении этой и других внешних программ к MathCAD можно прочитать в главе 7, “Работа с внешними источниками данных”.

Можно исследовать с помощью MathCAD задачу Коши (или начальную задачу) и краевую задачу для любых систем ДУ первого порядка. Очевидно, что дифференциальное уравнение любого порядка можно привести к системе уравнений первого порядка простой заменой переменных ($y_0(x)=y(x)$, $y_1(x)=y'(x)$, $y_2(x)=y''(x)$ …). Из ДУ в частных производных есть возможность решать только уравнения с двумя независимыми переменными: одномерные параболические и гиперболические уравнения, такие как уравнения теплопроводности, диффузии, волновые уравнения, а также двухмерные эллиптические уравнения (уравнения Пуассона и Лапласа).

В MathCAD нет универсальной функции для решения ДУ, а есть около двадцати функций для различных видов уравнений, дополнительных условий и методов решения.

Эти функции можно найти в библиотеке функций (напомним, что вызвать ее можно с помощью команды меню Insert⇨Function) в категории Differential Equation Solving (Решение дифференциальных уравнений).

Решение обычных дифференциальных уравнений и их систем

Решение ДУ в MathCAD возможно как с использованием группы решения (см. главу 4, раздел "Системы нелинейных уравнений"), так и без нее, с помощью одной функции, которой в виде аргументов передаются все параметры задачи. Оба способа обладают одинаковыми возможностями, и использование каждого зависит от конкретных целей (как и всегда, при использовании группы решения такая запись уравнений более привычна и наглядна для читателя, однако отдельная функция может быть использована в составе других функций и программ).

Использование группы решения

Для решения задачи Коши или краевой задачи с уравнением типа $y''(x) = f(y(x), y'(x), x)$ можно построить группу решения. Функцией, завершающей группу решения в задачах с обычными ДУ, является функция odesolve. На рис. 6.1 показан пример построения группы решения для решения задачи Коши с уравнением $y''(x) = \sin(y'(x) - x \cdot y(x) + x^2)$ и начальными условиями $y(0) = 1$ и $y'(0) = 0$ (два условия, поскольку уравнение второго порядка). Группа решения такой задачи состоит из следующих элементов.

1. Ключевое слово `given`.
2. Уравнение, решение которого нужно найти. Вы можете ввести любое уравнение вида $y''(x) = f(y(x), y'(x), x)$ (уравнения высших порядков и уравнения нелинейные по $y''(x)$ можно привести к системе ДУ первого порядка и решать способами, описанными далее в этой главе). Обратите внимание, что после каждого имени неизвестной функции должна быть введена в скобках независимая переменная (`x`).

Производные в уравнениях можно вводить с помощью обычного оператора производной, вставляемого с панели математических инструментов Calculus, а также с помощью штрихованной записи (рис. 6.1). Штрих вводится с помощью клавиатурного сочетания <Ctrl+F7>. Обратите внимание, что штрихи следует вводить между именем функции и независимой переменной в скобках.

3. Далее следует ввести дополнительные условия, накладываемые на решение, поскольку любое ДУ имеет бесконечное количество решений (или не имеет вообще). Количество дополнительных условий должно быть равно порядку уравнения, т.е. для уравнения второго порядка — два условия. Напомним, что при вводе уравнений и условий внутри группы решения следует пользоваться не обычным знаком равенства, а знаком, вводимым с помощью комбинации клавиш <Ctrl+=>.

Условия могут иметь вид: $y(a) = c_1$, $y'(a) = c_2$ в начальной точке для задачи Коши (смешанные условия вида $y(a) + y'(a) = c$ не поддерживаются в MathCAD). В случае краевой задачи нужно задать по одному такому условию на каждой границе интервала интегрирования.

4. Группа решения завершается функцией `odesolve (x,b)`. Первый аргумент функции — независимая переменная интегрирования. Очень важно, чтобы одно и то же имя переменной использовалось при введении уравнения и в функции `odesolve`. Второй аргумент — это верхняя граница отрезка интегрирования $[a, b]$. Функция `odesolve` возвращает решение задачи в виде функции. Конечно, эта функция не имеет символьного (аналитического) представления и может только вернуть численное значение решения уравнения в любой точке интервала $[a, b]$.

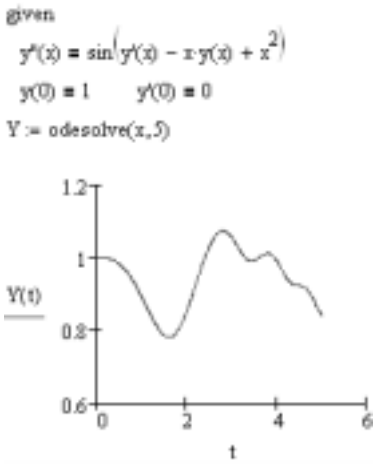


Рис. 6.1. Решение задачи Коши с помощью функции odesolve

Функция odesolve использует для решения уравнений метод Рунге-Кутты четвертого порядка с фиксированным шагом интегрирования. Если щелчком правой кнопки мыши на блоке формул с функцией `odesolve` вызвать соответствующее контекстное меню, то можно изменить метод вычисления решения, выбрав один из трех вариантов: Fixed — метод Рунге-Кутты с фиксированным шагом интегрирования (используется по умолчанию), Adaptive — также метод Рунге-Кутты, но с переменным шагом, изменяемым в зависимости от скорости изменения функции решения, Stiff — метод, адаптированный для решения жестких уравнений и систем (используется так называемый метод RADAU5). Различные виды уравнений и методы их численного исследования будут рассмотрены далее в этой главе.

Решение систем дифференциальных уравнений

Для того чтобы решить систему ДУ, можно, как и в задаче из предыдущего раздела, построить группу решения с функцией `odesolve`. Уравнения и дополнительные условия записываются после ключевого слова `given` с использованием знака равенства <Ctrl+=>. Очевидно, что в данном случае количество условий должно равняться сумме порядков всех уравнений, например для системы из двух уравнений первого порядка необходимо два дополнительных условия.

Лишь функция `odesolve` для системы уравнений имеет несколько иной, по сравнению с одним уравнением, синтаксис. Теперь она возвращает вектор функций, составляющих решение системы. Поэтому теперь в качестве первого аргумента функции `odesolve` нужно ввести вектор, состоящий из имен функций, использованных при вводе системы. Это нужно для того, чтобы конкретизировать их последовательность в век-

торе ответа. Второй и третий аргументы функции `odesolve` — независимая переменная системы и верхняя граница области интегрирования, т.е. то же самое, что и в задаче с одним уравнением. Иллюстрацию всего сказанного выше можно увидеть на рис. 6.2, где приведен пример решения задачи Коши с нелинейной системой из двух ДУ первого порядка.

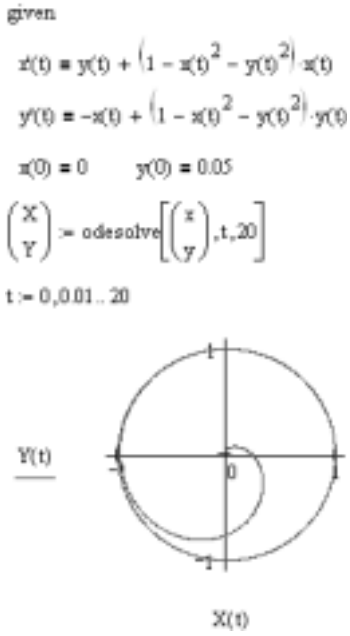


Рис. 6.2. Решение системы дифференциальных уравнений

Если вы попытаетесь вывести значение функции решения одного ДУ в любой точке вне интервала интегрирования, то результатом будет сообщение об ошибке, поскольку эта функция определена только на этом интервале. Но если то же самое попытаться сделать с решением системы уравнений, то вы получите какой-то результат. Но ни в коем случае не стоит воспринимать этот результат как решение системы ДУ в этой точке. А если вы действительно хотите получить решение в этой точке, увеличьте границу интервала интегрирования (последний аргумент функции `odesolve`).

Решение задачи Коши без использования группы решения

Кроме описанного выше способа, для решения задачи Коши с обычным ДУ или их системой можно обойтись и без группы решения. Это означает, что есть функции, которые через свои аргументы получают всю информацию о системе уравнений и начальных условиях и возвращают требуемое решение. Таких функций в MathCAD 12 несколько. Все они решают задачу Коши для системы ДУ первого порядка, но каждая из них использует для этого свой метод. Для простых систем не играет большой роли, какой метод использовать — все равно получите решение достаточно быстро и с высокой точностью. Но для сложных или специфических систем бывает, что некоторые методы вообще не могут дать удовлетворительного решения за приемлемое время. Именно для таких

сложных, но не редких случаев в MathCAD и введено несколько различных методов решения систем ДУ.

- `rkfixed` — метод Рунге-Кутты с фиксированным шагом интегрирования. Самый простой и быстрый метод, но далеко не всегда самый точный. Полностью аналогичен использованию функции `odesolve` с выбранным в контекстном меню методом Fixed.
- `Rkadapt` — метод Рунге-Кутты с переменным шагом интегрирования. Величина шага адаптируется к скорости изменения функции решения. Данный метод позволяет эффективно находить решения уравнений, в случае если они содержат как плавные, так и быстро меняющиеся участки. Полностью аналогичен использованию функции `odesolve` с выбранным в контекстном меню методом Adaptive.
- `Bulstoer` — метод Булирша-Штера. Этот метод более эффективен, чем метод Рунге-Кутты, в случае если решение является плавной функцией.

Обратите внимание, что имена функций `Rkadapt` и `Bulstoer` записаны с большой буквы. В MathCAD для некоторых имен функций неважно, с какой буквы они записаны, но для перечисленных функций это принципиально. На самом деле в MathCAD уже существуют функции с такими именами, но записаны они с маленькой буквы — `rkadapt` и `bulstoer`. Эти функции используются в тех случаях, когда важным является решение задачи в конечной точке интервала интегрирования. Их применение будет описано ниже в данной главе. Это же замечание касается и функций `Stiffr`, `Stiffb` и `Radau`, которые будут рассмотрены в следующем разделе.

Применение в документах любой из перечисленных функций выглядит сходным образом. Например, функция `Rkadapt` имеет следующий синтаксис: `Rkadapt(y,a,b,n,D)` (функции `rkfixed` и `Bulstoer` применяются абсолютно аналогично). Разберем назначение каждого аргумента по отдельности.

- `y` — вектор начальных значений неизвестных функций, входящих в систему. В случае одного уравнения и одной неизвестной функции — это просто число.
- `a` — начало отрезка, на котором ищется решение системы (отрезка интегрирования). Именно в этой точке значения неизвестных функций принимаются равными элементам вектора `y`.
- `b` — конец отрезка интегрирования. На рис. 6.3 выбрано $a = 0$, $b = 5$.
- `n` — количество частей, на которые разбивается отрезок $[a, b]$ при решении системы. Чем больше это число, тем точнее получается решение, но расчет занимает больше времени.
- `D(x,y)` — векторная функция, элементы которой содержат правые части уравнений системы в нормальной записи (когда левые части — первые производные от соответствующих функций, а в правых частях производные отсутствуют). Аргументами этой функции являются вектор `y`, элементы которого соответствуют различным неизвестным функциям системы, и скалярный аргумент `x`, соответствующий независимой переменной в системе. В случае одного уравнения функция `D` может быть скалярной функцией, зависящей от двух скалярных переменных `x` и `y`.

На рис. 6.3 с помощью функции `Rkadapt` решена та же задача, что и на рис. 6.1. Сравните эти два решения для того, чтобы лучше понять назначение аргументов `Rkadapt` и матрицы, которую эта функция возвращает. Возвращаемым значением всех перечис-

ленных выше функций является матрица. Первый столбец этой матрицы — это точки, на которые разбивался отрезок $[a, b]$, а остальные столбцы — это значения функций системы в этих точках. Если в аргументе функции `Rkadapt` было указано количество отрезков $n = 50$, то матрица будет содержать 51 строку, что соответствует всем точкам, в которых были вычислены решения системы.

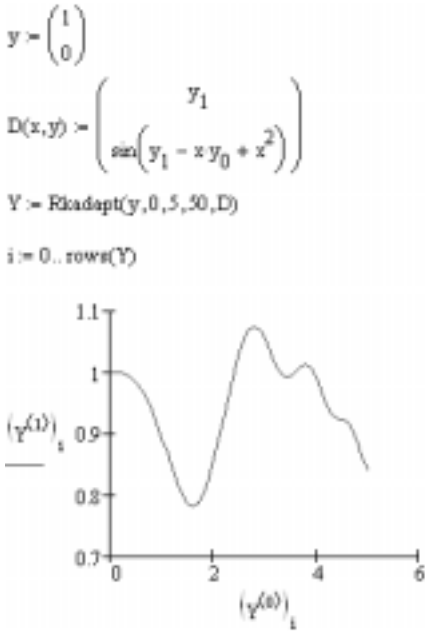


Рис. 6.3. Решение задачи Коши с помощью функции `Rkadapt`

Решение жестких уравнений и систем

Сложно дать математически точное определение жесткости, поскольку задачи, входящие в этот класс, весьма разнообразны. Чаще всего жесткими дифференциальными уравнениями называются уравнения, в решении которых есть плавно меняющаяся компонента, а также быстро затухающие возмущения.

Для жестких задач не работает обычный метод Рунге-Кутты или Булирша-Штера. Наличие быстро затухающего возмущения приводит к тому, что эти численные методы дают расходящееся решение. Для жестких задач разрабатываются специальные методы. Как уже упоминалось, в MathCAD предусмотрены три различные функции для решения жестких задач.

- `Radau` — метод RADAU5 для жестких систем. Полностью аналогичен использованию функции `odesolve` с выбранным в контекстном меню методом Stiff.
- `Stiffb` — метод Булирша-Штера, адаптированный для жестких систем.
- `Stiffr` — метод Розенброка.

Функция `Radau` имеет тот же синтаксис, что и функции `rkfixed`, `Rkadapt`, `Bulstoer` — `Radau(y,a,b,n,D)`. Все аргументы имеют то же назначение, и результат имеет ту же структуру, что было описано выше.

Использование функций `Stiffb` и `Stiffr` выглядит следующим образом: `Stiffb(y, a,b,n,D,J)`. Первые пять аргументов имеют то же назначение, что и в перечисленных выше функциях, но здесь есть еще один дополнительный аргумент — матричная функция `J(x,y)`. Аргументы этой функции те же, что и у функции `D`, а строится она следующим образом:

- для системы из n уравнений матрица J состоит из n строк и $n+1$ столбцов;
- $\boldsymbol{J}_{i,0} = \frac{d}{dx}\boldsymbol{D}_i(x,\boldsymbol{y})$ — элементы первого столбца (столбца с индексом 0) матрицы J — производные от элементов вектора $D(x,y)$ по независимой переменной (x);
- $\boldsymbol{J}_{i,j+1} = \frac{d}{d\boldsymbol{y}_j}\boldsymbol{D}_i(x,\boldsymbol{y})$ — оставшаяся часть матрицы J (квадратная матрица $n \times n$) — матрица Якоби системы уравнений.

На рис. 6.4 показан пример использования функций `Radau` и `Stiffb` для задачи Коши с уравнением вида: $y'(x) = L \cdot y(x) + \cos(x) - L \cdot \sin(x)$. Такая задача может быть решена аналитически, и ее общее решение имеет следующий вид: $y(x) = \sin(x) + y(0) \cdot e^{L \cdot x}$. Если в этой задаче выбрать верхнюю границу отрезка интегрирования b одного порядка с периодом изменения функции $\sin(x)$ (например, $b = 10$, см. рис. 6.4), а параметр L очень большим отрицательным числом (так, чтобы выполнялось условие $b >> -L^{-1}$), то задача становится жесткой, и обычные численные методы не могут найти ее решение.

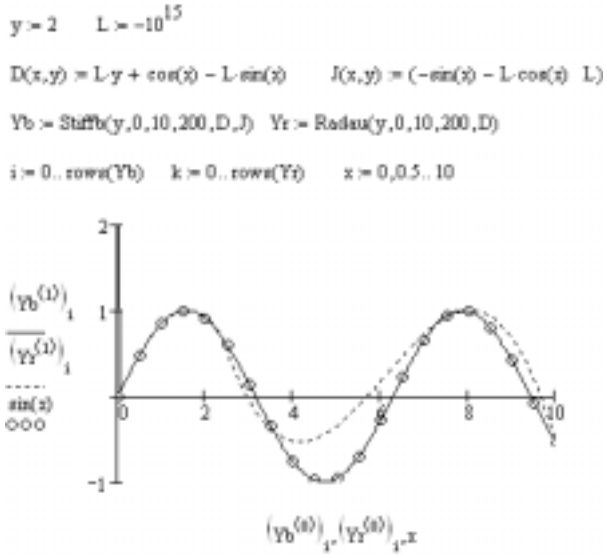


Рис. 6.4. Сравнение функций `Radau` и `Stiffb` на примере простой жесткой задачи

Как видно из рис. 6.4, функция `Radau` в такой задаче не дает точного решения на всем отрезке интегрирования. Конечно, это не означает, что среди трех перечисленных функций какие-то лучше решают жесткие задачи, а какие-то хуже. Просто для каждой конкретной задачи оптимальным будет свой метод решения. Именно поэтому в MathCAD их предусмотрено такое большое количество.

Линейная система ДУ первого порядка является жесткой, если соответствующая матрица Якоби содержит сильно различающиеся по величине собственные значения. На рис. 6.5 показан пример решения такой системы с помощью функций `Stiffr` и `Radau`. Как видим, для такой задачи все функции (функцию `Stiffb` попробуйте применить к этой задаче самостоятельно) дают точный ответ (на рис. 6.4 и 6.5 для сравнения приведены графики гладких компонент точного решения соответствующих задач).

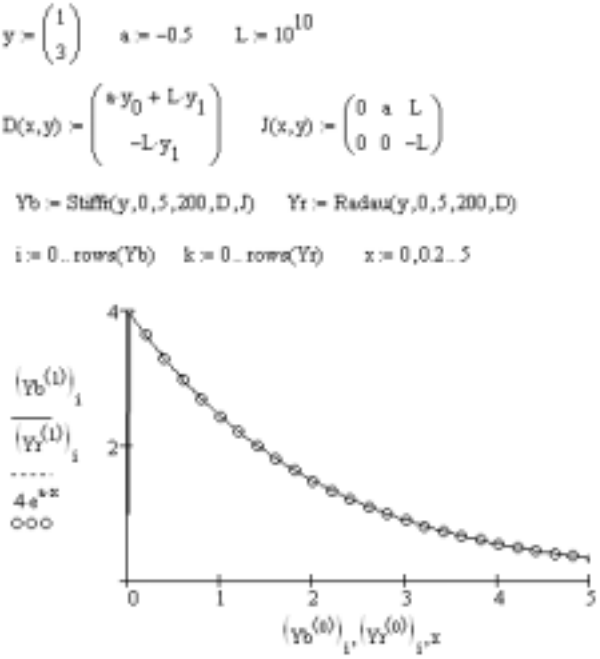


Рис. 6.5. Решение жесткой системы линейных уравнений

Конечно, точность решения, выдаваемого различными функциями, будет изменяться в зависимости от величины параметра L, определяющего, насколько жесткой является задача. Попробуйте, постепенно увеличивая (по модулю) параметр L, проверять решения, выдаваемые всеми описанными функциями. Общие тенденции таковы:

- при значениях $|L| < 10^3$ все функции выдают точные решения;
- при $|L| > 10^3 \div 10^6$ перестают работать функции `rkfixed`, `Rkadapt`, `Bulstoer` (система становится жесткой). Функции, предназначенные для жестких систем, продолжают выдавать решения. Точность этих решений зависит от выбранной функции и, конечно, от решаемой задачи;
- при $|L| > 10^{13}$ перестает работать функция `Stiffr`. Для более жестких задач следует применять функции `Radau` и `Stiffb`.

Конечно, в реальных задачах вы не будете заранее знать точного решения и не сможете так просто определить, насколько точен результат, выданный той или иной функцией (иногда для сложной задачи непросто даже определить, является ли она жесткой). Единственный совет, который здесь можно дать, — пробуйте различные функции, а для оценки точности результата можно просто подставить полученное решение в уравнение (существуют и другие методы контроля точности, но их описание выходит за рам-

ки данной книги). Если задача является жесткой, то не поленитесь вычислить матрицу Якоби (конечно, если это возможно) и воспользоваться функциями `Stiffb` или `Stiffr`, которые часто (хоть и не всегда) оказываются более точными, чем `Radau`.

Получение решения задачи Коши на верхней границе интервала интегрирования

Как уже было сказано, в MathCAD также существуют пять функций: `rkadapt`, `bulstoer`, `radau`, `stiffb` и `stiffr`, действия которых отличны от соответствующих функций, записанных с большой буквы. Эти функции следует использовать в тех случаях, когда интерес представляет решение не на всем интервале интегрирования, а только в конечной точке интервала. Перечисленные функции дают возможность находить это значение с высокой и контролируемой точностью. Использование этих функций на примере функции `bulstoer` и той же задачи, что решалась ранее, показано на рис. 6.6. Эта функция возвращает матрицу, структура которой уже была описана ранее: в первом столбце содержатся значения независимой переменной, а в остальных — соответствующие значения неизвестных функций системы. Использование функции `bulstoer`, как и остальных вышеперечисленных функций, выглядит следующим образом: `bulstoer(y, a,b,acc,D,kmax,s)`. Рассмотрим, как и ранее, назначение каждого из аргументов по отдельности (рис. 6.6).

$$y := \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$D(x,y) := \begin{pmatrix} y_1 \\ \sin\left(y_1 - x\,y_0 + x^2\right) \end{pmatrix}$$

$$Y := \text{bulstoer}\left(y,0,5,10^{-5},D,10,0.1\right)$$

$$Y_{\text{rows}(Y)-1,1} = 0.838438557519705$$

$$Y := \text{bulstoer}\left(y,0,5,10^{-16},D,10,0.1\right)$$

$$Y_{\text{rows}(Y)-1,1} = 0.838424496672918$$

$$Y := \text{Bulstoer}(y,0,5,100,D)$$

$$Y_{\text{rows}(Y)-1,1} = 0.838424738962545$$

$$Y := \text{Bulstoer}(y,0,5,10000,D)$$

$$Y_{\text{rows}(Y)-1,1} = 0.838424496672383$$

Рис. 6.6. Точность функций `Bulstoer` и `bulstoer` на конце интервала интегрирования

- `y` — вектор значений неизвестных функций системы в начальной точке отрезка интегрирования.
- `a` — начало отрезка интегрирования.

- `b` — конечная точка отрезка интегрирования. Если вы еще не забыли материал предыдущих двух разделов, то уже заметили, что эти же первые три аргумента задавались и для соответствующих функций, записанных с большой буквы.
- `acc` — величина, задающая точность, с которой необходимо найти решение в конечной точке интервала интегрирования. Естественно, во всех остальных точках интервала точность будет намного ниже.
- `D` — уже знакомая функция $D(x,y)$, содержащая правые части уравнений системы в нормальной записи.
- `kmax` — максимальное количество интервалов, на которые будет разбит отрезок интегрирования. Рассматриваемые функции разбивают отрезок интегрирования не на равные интервалы, а таким образом, чтобы получить точное решение в конечной точке. Зачастую для этого достаточно небольшого количества интервалов.
- `s` — минимальный шаг между точками. Этот параметр накладывает нижнюю границу на длину интервалов. Два последних аргумента никак не влияют на точность вычисления значений в конечной точке отрезка интегрирования. Подбором этих параметров можно добиться приемлемой точности вычисления внутри интервала интегрирования. Такая возможность может пригодиться, если вы, например, хотите вычислить точное значение функции в конечной точке интервала интегрирования и построить график поведения функции на всем интервале.

На рис. 6.6 видны преимущества функции bulstoer в том случае, когда необходимо знать решение только в конечной точке отрезка интегрирования:

- вы всегда знаете заранее, с какой точностью получен результат (в конечной точке);
- скорость работы функции `bulstoer` в среднем намного выше, чем функции `Bulstoer`, поскольку точность вычислений внутри интервала очень низкая.

Приведение краевых задач к задачам Коши

Как вы, наверное, уже заметили, в MathCAD все функции предназначены только для решения задачи Коши. И действительно, для решения краевой задачи в MathCAD нет отдельной функции. Однако есть функции, позволяющие превратить краевую задачу в задачу Коши. Эти функции "угадывают" недостающие начальные условия исходя из того, что решение должно удовлетворять заданным условиям в конечной точке интервала интегрирования. Простейшей из функций, предназначенных для приведения краевой задачи к задаче Коши, является `sbval`. Для того чтобы решить двухточечную краевую задачу с помощью этой функции, нужно выполнить следующие действия (рис. 6.7).

1. Задайте вектор `v` с количеством элементов равным количеству недостающих начальных условий. Значения элементов этого вектора — это начальные приближения, исходя из которых будет происходить поиск недостающих начальных условий. На данном этапе не конкретизируется, какой из элементов вектора будет соответствовать начальному значению той или иной неизвестной функции в задаче.
2. Задайте функцию `D(x,y)`. Эта функция уже описывалась выше. Она представляет собой вектор, каждый элемент которого — это правая часть одного из уравнений системы.

3. Задайте еще одну векторную функцию `load(x,v)`. Это функция от скалярного аргумента `x` и вектора `v`, который имеет столько же компонент, сколько недостающих начальных условий в системе. Сам вектор `load` должен содержать такое же количество элементов, как и вектор `D`, т.е. столько, сколько должно быть начальных условий в задаче. Если начальное значение какой-либо из функций известно, то соответствующий элемент вектора `load` должен содержать это значение. Для функций, начальное значение которых не известно, соответствующий элемент вектора `load` должен содержать один из элементов вектора `v`.
4. Следует задать еще одну векторную функцию `score(x,y)`. Аргументы этой функции — скаляр `x` и вектор `y`, который имеет столько элементов, сколько уравнений в системе. Количество компонент вектора `score` должно равняться количеству граничных условий, заданных в конечной точке отрезка интегрирования. На самом деле каждая компонента этого вектора задает одно из граничных условий в конечной точке. Например, если в задаче есть граничное условие $\boldsymbol{y}_i(b) = c$, то один из элементов вектора `score` должен быть функцией, которая обращается в нуль при значениях $x = b$ и $\boldsymbol{y}_i(b) = c$. Конкретный вид этой функции не играет особой роли, поэтому проще всего задавать ее в таком виде: $\mathtt{score}_k\mathtt{(x,y)} := \mathtt{y}_i\mathtt{-c}$. Таким же образом должны быть заданы все элементы вектора `score` для всех конечных условий задачи.
5. Теперь все введенные величины нужно использовать как аргументы в функции `sbval`. Использование этой функции выглядит следующим образом: `Y := sbval(v, a, b, D, load, score)`. Аргументы `a` и `b` — это начало и конец отрезка интегрирования, а все остальные аргументы были описаны выше.
6. Результатом функции `sbval` будет вектор, содержащий недостающие начальные значения. Их последовательность задается той последовательностью, в которой были использованы компоненты вектора `v` в функции `load`. Постройте вектор начальных значений, используя известные начальные значения, а также элементы вектора `Y`, там, где значения были неизвестны (рис. 6.7).
7. Теперь можно решать полученную задачу как задачу Коши с помощью, например, функции `Rkadapt`, что и сделано на рис. 6.7. Также на рис. 6.7 показано, что полученное решение действительно удовлетворяет конечному условию исходной задачи.

$$v_0 := 10 \qquad D(x,y) := \begin{pmatrix} y_1 \\ y_1 - 3x\,y_0 \end{pmatrix}$$

$$load(x,v) := \begin{pmatrix} 1 \\ v_0 \end{pmatrix} \qquad score(x,y) := y_0 - 4$$

$$Y := sbval(v,0,3,D,load,score) \qquad Y = (-2.908)$$

$$y := \begin{pmatrix} 1 \\ Y_0 \end{pmatrix}$$

$$Z := Rkadapt(y,0,3,100,D) \qquad \left(Z^{\langle 1 \rangle}\right)_{rows(Z)-1} = 4$$

Рис. 6.7. Решение краевой задачи с помощью функций `sbval` и `Rkadapt`

Решение уравнений в частных производных

Как уже упоминалось в начале главы, в системе MathCAD есть возможность решать ДУ в частных производных и их системы. Средства MathCAD позволяют решать одномерные параболические и гиперболические уравнения (с одной пространственной и одной временной переменной), а также двухмерное уравнение Пуассона. Такой, казалось бы, узкий круг решаемых задач на самом деле охватывает подавляющее большинство задач, возникающих в физике и технике.

Использование группы решения для дифференциальных уравнений параболического типа

Для решения дифференциальных уравнений в частных производных параболического типа можно построить группу решения с функцией `pdesolve`. Такая группа решения состоит из следующих элементов (рис. 6.8).

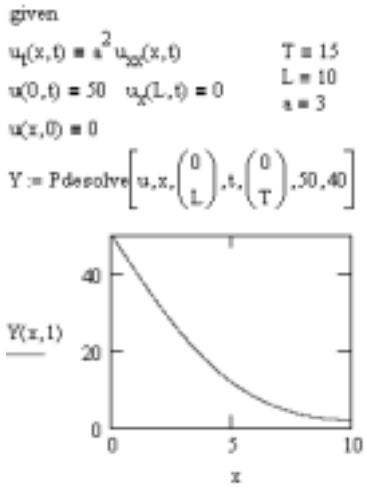


Рис. 6.8. Решение одномерной задачи теплопроводности

1. Ключевое слово `given`, чтобы сигнализировать о начале группы решения.
2. Уравнение, которое нужно решить. Уравнение должно иметь такой вид: $u_t(x, t) = f(x, t, u, u_x, u_{xx})$. Для ввода производных в данном случае нельзя использовать обычный оператор производной, а нужно пользоваться нижним индексом, как это обычно делается в литературе для записи уравнений в частных производных. При этом нижний индекс следует вводить не как элемент массива (клавиша < [>), а как текстовый нижний индекс (клавиша < . >). Для того чтобы ввести вторую производную от функции u по переменной x, следует набрать на клавиатуре: `u . x x`.
3. Граничные условия для функции $u(x, t)$. Если уравнение второго порядка по x, то и граничных условий должно быть два. Вы можете использовать как граничные условия Дирихле ($u(x_0, t) = w(t)$), так и граничные условия Неймана ($u_x(x_0, t) = w(t)$) или даже их комбинацию (что и было сделано в задаче на рис. 6.8).

4. Начальное значение для неизвестной функции — $u(x, 0)$.
5. Функция `pdesolve(u,x,xrange,t,trange,xpts,tpts)`. Ее аргументы имеют следующее назначение.
   - `u` — имя функции, относительно которой решается уравнение. Для системы уравнений здесь должен быть вектор имен функций (как в odesolve).
   - `x` — имя пространственной переменной.
   - `xrange` — двухкомпонентный вектор, задающий начало и конец интервала изменения пространственной переменной.
   - `t` — имя временной переменной. Основная разница между пространственной и временной переменными в данном случае — это то, что все уравнения могут содержать только первые производные по временной переменной.
   - `trange` — еще один двухкомпонентный вектор. Этот вектор задает начало и конец временного интервала, на котором решается задача.
   - `xpts`, `tpts` — количество точек, разбивающих для интегрирования пространственный и временной интервалы соответственно. Эти два параметра можно не указывать, тогда количество точек будет выбрано автоматически из соображений достаточной точности. Автор рекомендует задавать эти параметры во всех задачах, кроме самых простых, поскольку во многих случаях высокая точность вычислений теряет смысл из-за погрешности, вносимой самим методом.

Как пример решения уравнений с помощью функции `pdesolve`, на рис. 6.8 решена одномерная задача теплопроводности для однородного бруска, один конец которого теплоизолирован, а другой поддерживается при определенной температуре. На рис. 6.8 использован оператор глобального присваивания, который дает возможность использовать созданную таким образом переменную в любой точке документа. Напомним, что для ввода этого оператора можно воспользоваться клавишей < ~ >.

Решение дифференциальных уравнений гиперболического типа и систем

Функция `pdesolve` также позволяет решать системы ДУ в частных производных первого порядка по времени. Такая возможность может быть использована для решения задач с ДУ гиперболического типа. Ведь поскольку уравнения гиперболического типа содержат вторую производную по времени, то они не могут быть напрямую введены для решения функцией `pdesolve`. ДУ гиперболического типа должно быть приведено к системе из двух уравнений первого порядка по времени (как это делалось ранее для обычных ДУ высоких порядков). А далее полученная задача может быть решена с помощью функции `pdesolve` как система уравнений.

Пример на рис. 6.9 показывает, как привести уравнение гиперболического типа к системе из двух уравнений и решить полученную систему. Как видите, группа решения в таком случае строится точно так же, как и для одного уравнения. Для каждой неизвестной функции должно быть задано начальное значение. Количество граничных условий для каждой функции может быть различным: если в системе встречается вторая производная от данной функции по пространственной переменной, то нужно задать два граничных условия, если есть только первая производная — одно условие, если производных по пространственной переменной от данной функции в системе нет, то и граничных условий для нее не нужно (именно такая ситуация и реализуется для функции v в системе на рис. 6.9).

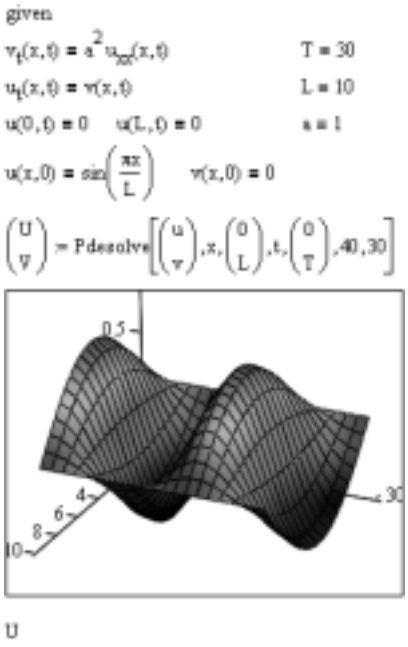


Рис. 6.9. Решение волнового уравнения с помощью группы решения

Использование функции numol

В MathCAD есть возможность решать уравнения и системы параболического и гиперболического типов без создания группы решения (конечно, уравнения гиперболического типа должны быть приведены к системе уравнений первого порядка по времени). Для этого служит встроенная функция `numol`. Ее использование выглядит следующим образом (рис. 6.10).

1. Постройте векторную функцию `pdef(x,t,u,`u_x`,`u_{xx}`)`. Эта функция — аналог функции `D(x,y)`, которую строили при решении систем обычных ДУ. Количество элементов этого вектора равно сумме *npde* + *npae*. Здесь *npde* — количество ДУ в частных производных, которые входят в систему: *npae* — количество алгебраических условий, связывающих между собой неизвестные функции. Первые *npde* элементов вектора `pdef` задают правые части дифференциальных уравнений (подразумевается, что в левую часть уравнения мы перенесли производные по времени). Остальные элементы задают алгебраические уравнения, связывающие между собой неизвестные функции (в этих элементах подразумевается, что левая часть равна нулю).
2. Задайте начальные условия для неизвестных функций в виде векторной функции `init(x)`. Каждый элемент этого вектора должен содержать начальное условие для соответствующей функции.
3. Задайте граничные условия в виде матричной функции `bc(t)`. Эта матрица должна содержать три столбца и столько строк, сколько функций в системе. В первых двух столбцах этой матрицы задаются условия, накладываемые на ту или иную функцию в начальной и конечной точках отрезка интегрирования соответственно.

Третий столбец определяет тип условий. Если для функции задаются условия Дирихле, то в третьем столбце соответствующей строки нужно ввести "D", а если условия Неймана — "N". Если в системе отсутствует вторая производная от той или иной функции по пространственной переменной, тогда для нее нужно задавать только одно граничное условие, а вместо второго ввести "NA" (при этом в третьем столбце обязательно должно быть введено "D"). Если же для какой-то функции в системе вовсе отсутствуют производные по пространственной переменной, то граничные условия для нее не нужны (соответствующая строка в матрице bc должна иметь вид: "NA", "NA", "D").

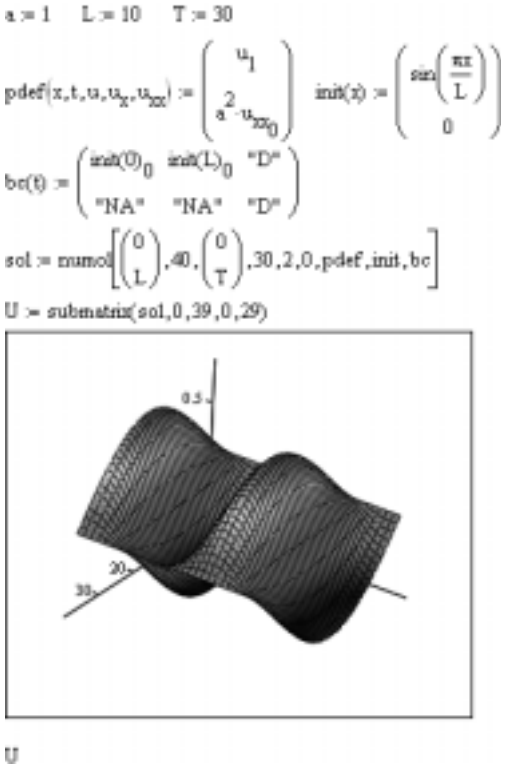


Рис. 6.10. Решение волнового уравнения с помощью функции numol

Обратите внимание, что в функции numol вы не сможете использовать смешанные граничные условия, когда на одной границе интервала задается значение самой функции, а на другой — ее производной. Для решения, например, такой задачи, как на рис. 6.8, можно использовать только функцию pdesolve.

4. Теперь можно решать введенную задачу. Для этого можно ввести: sol := = numol(xend,xpts,tend,tpts,npde,npae,pdef,init,bc). Разберем последовательно назначение тех ее аргументов, которые еще не были описаны:
   - xend — двухкомпонентный вектор, элементы которого задают начало и конец пространственного отрезка интегрирования;
   - xpts — количество точек, разбивающих пространственный интервал;

- `tend` — также двухкомпонентный вектор, но он задает границы временного интервала;
- `tpts` — количество точек, разбивающих временной интервал;
- `npde`, `npae` — эти числа задают количество уравнений и алгебраических условий в системе (они были описаны выше).

Результат функции `numol` при решении системы уравнений — матрица, состоящая из *xpts* строк и *tpts*·(*npde* + *npae*) столбцов. Эта матрица объединяет несколько матриц размера *xpts* × *tpts*, каждая из которых содержит значения одной из неизвестных функций системы. Как видно из рис. 6.10, решение для любой функции может быть легко выделено из результата `numol` с помощью функции `submatrix`.

Решение уравнения Пуассона с нулевыми граничными условиями

Как уже было сказано, описанные выше функции не дают возможности решать уравнения эллиптического типа. Но для одного из уравнений эллиптического типа, а именно двухмерного уравнения Пуассона, в MathCAD есть отдельные встроенные функции: `multigrid` и `relax`.

Если требуется решить уравнение Пуассона с нулевыми граничными условиями (функция равняется нулю на всех границах области), то для этого можно воспользоваться функцией `multigrid` (рис. 6.11). Для того чтобы воспользоваться этой функцией, нужно предварительно задать квадратную матрицу `M` размера $2^n + 1$, где n — любое целое положительное число. Такой размер матрицы требуется для правильной работы алгоритма `multigrid`. Каждый элемент матрицы `M` — это число, задающее интенсивность источника в данной точке квадратной области интегрирования.

Вызов функции `multigrid` выглядит следующим образом: `U := multigrid(M, num)`. Здесь параметр `num` задает количество циклов в каждой итерации. Значение `num := 2` даст хорошую точность в большинстве задач, а скорость вычислений от этого параметра почти не зависит.

Результатом функции `multigrid` является матрица того же размера, что и `M`, содержащая решение уравнения во всех точках области. Точность вычислений в данном случае никак не контролируется и определяется только размерами матрицы `M` — чем мельче вы разобьете область интегрирования, тем точнее будет результат.

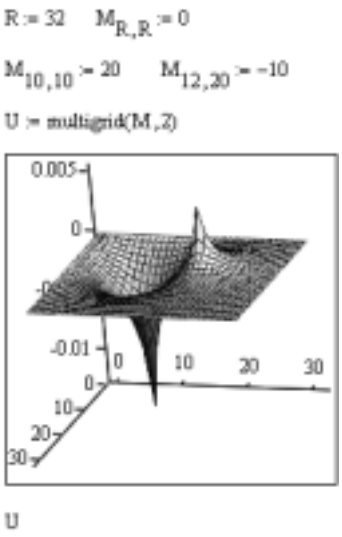


Рис. 6.11. Решение уравнения Пуассона с нулевыми граничными условиями

Решение уравнения Пуассона с ненулевыми граничными условиями

Для решения уравнения Пуассона с ненулевыми граничными условиями, а также уравнения Лапласа следует использовать функцию `relax` (рис. 6.12). Использование данной функции подразумевает следующую последовательность действий.

1. Задайте пять квадратных матриц a, b, c, d, e. Эти матрицы будут служить коэффициентами в формуле приближенного вычисления Лапласиана: $(\Delta u)_{ij} = a_{ij} \cdot u_{i+1j} + b_{ij} \cdot u_{i-1j} + c_{ij} \cdot u_{ij+1} + d_{ij} \cdot u_{ij-1} + e_{ij} \cdot u_{ij}$. Стандартные значения для элементов этих матриц: $\boldsymbol{a}_{ij} = \boldsymbol{b}_{ij} = \boldsymbol{c}_{ij} = \boldsymbol{d}_{ij} = 1$, $\boldsymbol{e}_{ij} = -4$. Размер этих матриц может быть выбран любой. Главное, чтобы все матрицы, задаваемые для функции relax, были одинакового размера.
2. Задайте матрицу s, задающую интенсивность источника в каждой точке квадратной области. Если все элементы этой матрицы имеют нулевые значения, то полученный результат будет решением уравнения Лапласа.
3. Задайте матрицу f. Первый и последний столбцы и первая и последняя строки этой матрицы задают граничные условия для решения уравнения. Значения внутренних элементов матрицы не играют особой роли, а используются лишь как начальное приближение при поиске решения.
4. Теперь можно использовать функцию relax. Это делается следующим образом: U := relax(a,b,c,d,e,s,f,r). Здесь r — так называемый спектральный радиус Якоби. Это число в диапазоне от 0 до 1. Если функция relax не может решить уравнение, попробуйте уменьшить значение спектрального радиуса.

$R := 20 \quad i := 0..R \quad j := 0..R$

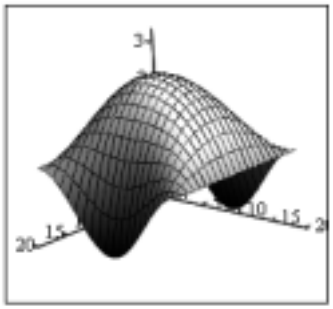
$a_{i,j} := 1 \quad b := a \quad c := a \quad d := a \quad e := -4a$

$s_{i,j} := -0.1 \quad r := 1 - \frac{\pi}{R}$

$f_{0,j} := \cos\left(\pi \cdot \frac{j}{10}\right) \quad f_{R,j} := \cos\left(\pi \cdot \frac{j}{10}\right)$

$f_{i,0} := 1 \quad f_{i,R} := 1$

U := relax(a,b,c,d,e,s,f,r)



U

Рис. 6.12. Решение уравнения Пуассона с помощью функции relax

Для функции relax, как и для multigrid, единственным параметром, контролирующим точность, является размер используемых матриц. Также при использовании функции relax не следует забывать о том, что все матрицы должны быть квадратными и их размеры должны совпадать.

Резюме

В данной главе мы познакомились со средствами решения дифференциальных уравнений в среде MathCAD. Для решения задачи Коши или краевой задачи с обычным ДУ или системой обычных ДУ можно построить группу решения с функцией `odesolve`. Преимущество использования группы решения состоит в том, что условие задачи записано в общепринятой форме, и документ легко читается даже человеком, не знакомым с MathCAD.

Для решения задачи Коши без использования группы решения в MathCAD предусмотрено несколько функций: `rkfixed`, `Rkadapt`, `Bulstoer`, `Radau`, `Stiffr`, `Stiffb`. Каждая из этих функций реализует свой метод решения и предназначена для своего класса задач. Например, последние три функции предназначены для решения так называемых жестких задач. Решение таких задач является суперпозицией гладкой, медленно меняющейся функции и быстро затухающего возмущения.

Когда интерес представляет решение задачи только в конечной точке интервала интегрирования, уместно воспользоваться функциями `rkadapt`, `bulstoer`, `radau`, `stiffr`, `stiffb`. Эти функции хоть и не дают высокой точности внутри интервала интегрирования, зато позволяют получить значение в конечной точке с любой заданной точностью.

Двухточечная краевая задача не может быть напрямую решена средствами MathCAD (без использования группы решения). Но есть функция `sbval`, которая позволяет привести краевую задачу к задаче Коши. Эта функция "угадывает" недостающие начальные условия таким образом, чтобы решение удовлетворяло условиям в конечной точке интервала.

Кроме обычных ДУ и их систем, MathCAD позволяет также решать некоторые ДУ в частных производных. Для решения одномерных (с одной пространственной и одной часовой переменной) ДУ параболического или гиперболического типа можно построить группу решения с функцией `pdesolve`. То же самое, но без группы решения можно сделать с помощью функции `numol`.

Перечисленные выше функции не могут решать уравнения эллиптического типа. Для этих уравнений, а точнее для уравнения Пуассона, в MathCAD предусмотрены две функции: `multigrid` и `relax`. Первая из них используется в случае нулевых граничных условий, вторая — в случае ненулевых.

Тесты

Эти тесты помогут вам закрепить материал данной главы. Ответы ищите в приложении А.

Найдите правильный ответ

Каждый из предложенных вопросов может иметь несколько правильных ответов.

1. Какие из перечисленных классов ДУ можно решать средствами MathCAD:
   а) задача Коши с обычным ДУ второго порядка;
   б) краевая задача с обычным ДУ третьего порядка;
   в) ДУ в частных производных гиперболического типа;
   г) уравнение Пуассона.

2. Какие из перечисленных функций не подходят для решения жестких задач:
   а) `Stiffr`;
   б) `stiffr`;
   в) `Radau`;
   г) `rkfixed`.
3. Какие из перечисленных функций используются в составе группы решения:
   а) `Radau`;
   б) `odesolve`;
   в) `numol`;
   г) `relax`;
   д) `pdesolve`.

Правда или ложь?

Каждое утверждение либо верно, либо нет.

4. MathCAD позволяет решать обычные ДУ любого порядка.
5. Функция `rkadapt` возвращает решение системы только в конечной точке интервала интегрирования.
6. В MathCAD невозможно решить краевую задачу без использования группы решения.
7. Последний аргумент функции `Stiffr` — матрица `J` (это матрица Якоби системы).
8. Размер матрицы в функции `multigrid` может быть только $2^n + 1$, где n — целое положительное число.