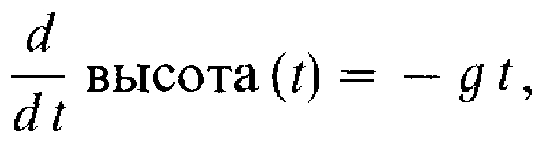
**Раздел 6. Обыкновенные дифференциальные уравнения**

Основная терминология. Устойчивые и неустойчивые уравнения и численные методы. Жесткие дифференциальные уравнения. Метод Эйлера. Точность и устойчивость численных методов. Порядок метода интегрирования. Неявные методы. Многошаговые методы. Порядок и погрешность многошаговых методов. Устойчивость многошаговых методов. Метод функциональной итерации и метод Ньютона для решения неявных уравнений. Многозначные методы. Некоторые другие многозначные методы. Связь многошаговых и многозначных методов

8.1. **Введение (320)**

Дифференциальное уравнение (ДУ) описывает, как изменяется функция. Это позволяет нам моделировать движения и процессы, непрерывно меняющиеся во времени. В качестве простого примера допустим, что с крыши здания брошен мяч. Тогда высота, на которой находится мяч, удовлетворяет ДУ



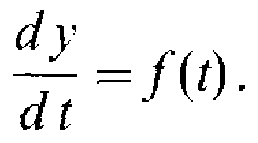
где *g* – ускорение свободного падения, а переменная *t* представляет собой время, отсчитываемое от начала падения мяча. Это уравнение можно решить, интегрируя его правую часть, и получить



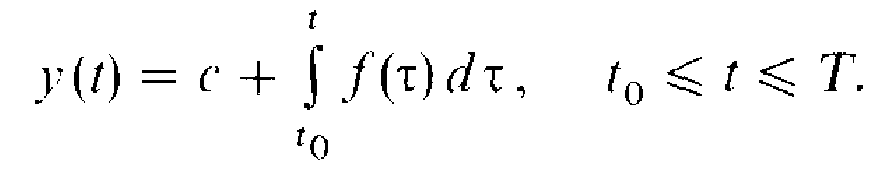
Отметим несколько обстоятельств. Во-первых, решение зависит от высоты здания, т.е. от величины высота(0), и, следовательно, не определяется однозначно ДУ. Во-вторых, решением является не число, а функция.

Математические модели, основанные на ДУ, широко применяются во многих научных дисциплинах для описания окружающего нас мира. От анализа химических реакций на атомарном уровне до глобального предсказания погоды, изучения комет, планет и галактик – всюду ДУ играют важную роль. Даже системы, которые по своей природе дискретны, такие, как численность кроликов в биологической экосистеме, часто хорошо аппроксимируются ДУ. В этом параграфе вводятся некоторые основные идеи, необходимые для понимания обыкновенных дифференциальных уравнений (ОДУ). В последующих параграфах мы вернемся к каждой из этих идей и рассмотрим их более подробно.

*ОДУ* – это уравнение, содержащее функцию и ее производные. Простейшее ОДУ имеет вид



В примере с мячом *y* – это высота, *f*(*t*) = – *gt*. Пусть нас интересует решение начиная с некоторого *t* = *t*0. Если функция *f*(*t*) непрерывна на отрезке [*t*0, *T*], то решение рассматриваемого уравнения записывается в виде



функция *y*(*t*) определена с точностью до константы. В нашем примере константа с равна начальной высоте, т.е. величине высота(0). Таким образом, наше ОДУ имеет семейство решений, каждое из которых отличается от другого на аддитивную константу. Вид данного уравнения таков, что в фиксированной точке *t* все кривые из семейства решений имеют одинаковый наклон *dy*/*dt*. Чтобы выделить единственную кривую, возьмем то решение, которое принимает заданное значение *А* в точке *t* = *t*0. Величина *А* определяется конкретной постановкой задачи; в нашем примере она выражает высоту здания.

Простейшее уравнение иллюстрирует тесную связь между задачами интегрирования и решения ДУ. Однако есть и определенные различия. Одно из важнейших различий в том, что при интегрировании результатом является единственное число – величина интеграла. Здесь же ищется функция, определяемая на отрезке. Иногда решение ДУ требуется определить только в одной точке, например, узнать высоту мяча на третьей секунде падения. Тогда, как и при интегрировании, результатом будет число. Однако гораздо чаще ищется решение на отрезке. Так, если мы хотим выразить зависимость высоты мяча от времени, то лучше искать решение в виде функции. Явную формулу для решения удается выписать только в простейших случаях, а в общем случае это представляется невозможным или слишком сложным. Другое различие между ОДУ и интегралами заключается в том, что первые одномерны, т.е. зависят от одного независимого переменного, а интегралы часто бывают многомерными.

Поскольку решением ОДУ является функция, следует прийти к соглашению, какой смысл мы будем вкладывать в выражение «решить уравнение на компьютере». Общепринятый подход заключается в дискретизации задачи. Это значит, что решение *y*(*t*) вычисляется только в конечном числе дискретных точек (узлов сетки), а не на всем непрерывном отрезке. Например, если [*t*0, *T*] = [0, 1], то решение можно вычислить в узлах 0.0, 0.01, 0.02, ..., 0.99, 1.0. Значение в какой-либо промежуточной точке можно получить интерполированием. Узлы сетки не обязательно размещать равномерно. Бывает, что узлы задаются заранее, поскольку требуется найти решение в определенных точках или построить график или таблицу по специальным наборам узлов. Однако лучше позволить программе определить узлы самостоятельно, поскольку, учитывая информацию о конкретном ДУ, удается вычислить приближенное решение точнее и эффективнее. Например, если решение сильно меняется в некоторой подобласти, то необходимо разместить в такой подобласти много узлов сетки, чтобы отследить изменение решения. В этом отношении задачи численного решения ОДУ и интегрирования схожи.

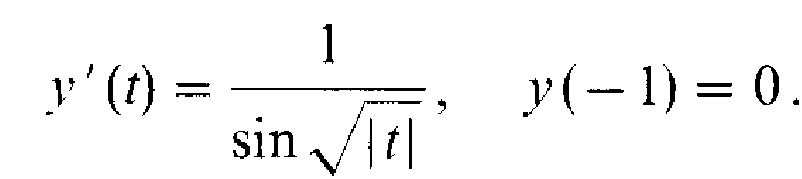
Существует несколько источников возникновения ошибок при решении ДУ. Так, ошибки появляются при дискретизации дифференциальной задачи, подобно случаю разностной аппроксимации производной. Во многих, но, к сожалению, не во всех случаях повторное решение задачи на более частой сетке позволяет добиться большей точности. Однако иногда малые ошибки, внесенные в начале вычислений, могут совершенно исказить решение, если только не подобрать подходящий численный метод. Это явление иногда называется «неустойчивостью». Далее, вычислениям, как правило, сопутствуют ошибки округления. В численных методах приходится идти на компромисс, сочетая стремление уменьшить отмеченные виды ошибок с необходимостью решить задачу с меньшими вычислительными затратами.

Часто возникает потребность в решении систем ОДУ, содержащих несколько неизвестных функций. Многие методы решения одного уравнения можно применить и к системе уравнений, хотя реализация такого алгоритма может оказаться утомительной и труднее будет использовать геометрическую интуицию. ДУ высокого порядка, содержащие вторую, третью и более высокие производные, можно записать в виде системы дифференциальных уравнений первого порядка, поэтому нет особых причин рассматривать уравнения порядка выше первого.

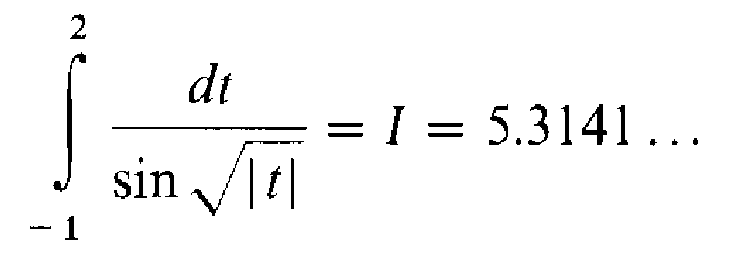
В том случае, когда дифференциальная задача описывает два взаимодействующих процесса, один из которых претерпевает быстрые изменения, а другой – медленные, могут возникнуть определенные затруднения. Примером может служить модель удара по барабану, в которой соседствуют быстрые колебания, отвечающие тону инструмента, и медленное ослабление силы звука. Чтобы отследить быстрые колебания, потребуется взять частую сетку, что приведет к большому объему вычислений. Однако уже через весьма малое время после удара вклад быстрых колебаний в решение уменьшается, поэтому для расчета силы звука можно было бы обойтись редкой сеткой. Задачи такого рода принято называть жесткими. Некоторые численные методы неприемлемы для решения жестких задач, ибо они неуклонно стремятся следовать за быстрыми движениями, даже если вклад последних не столь важен, как общий характер решения. Для решения жестких задач разработаны специальные алгоритмы; некоторые из них мы рассмотрим ниже.

В этой книге мы обсудим только начальные задачи для ОДУ. Под начальной задачей (НЗ) мы подразумеваем ДУ, решение которого в некоторой точке (обозначеной выше через *t*0) дано, а при *t*, б***о***льших или меньших, чем *t*0, его требуется найти.

Численные методы решения начальных задач стартуют из начальной точки и стараются отследить или «учуять» кривую решения. Часто они называются *маршевыми* методами. Чтобы решить задачу, в любом методе приходится вычислять значения правой части уравнения *f*. Если в какой-либо точке *f* обращается в бесконечность, то метод может не сработать. Такое может случиться, например, в окрестности точки *t* = 0 при решении задачи



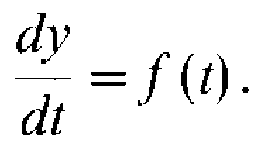
Тем не менее соответствующий интеграл



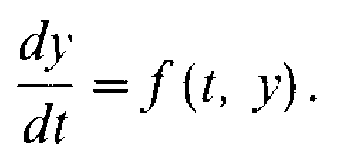
нетрудно вычислить с помощью современных программ интегрирования. В этом заключается еще одно различие между ДУ и интегралами. Задача 8.1 дает хорошую возможность продолжить сопоставление начальных задач и квадратур.

8.1.1. **Основная терминология (323)**

Как отмечено выше, простейшее ОДУ записывается в виде



Это ОДУ *первого порядка*, поскольку наивысший порядок производных, содержащихся в уравнении, равен единице. Решение этого уравнения можно найти, интегрируя правую часть, и все решения будут различаться на аддитивную константу. Рассмотрим ОДУ первого порядка более общего вида



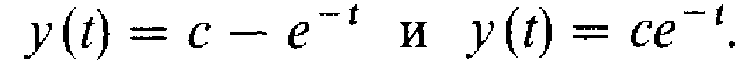
Это уравнение также имеет семейство решений (при соответствующих предположениях об *f*), но отдельные решения различаются уже не на аддитивную константу, поскольку при фиксированном *t* каждое из решений имеет свой наклон.

**Пример 8.1. Семейства решений. (324)**

Будем обозначать производную *dy/dt* через *y*′. Рассмотрим два разных уравнения



которые обладают соответственно решениями



Семейства решений этих уравнений представлены на рис. 8.1. Отметим, что при фиксированном *t* производная *y*' зависит от *y* во втором случае и не зависит в первом.

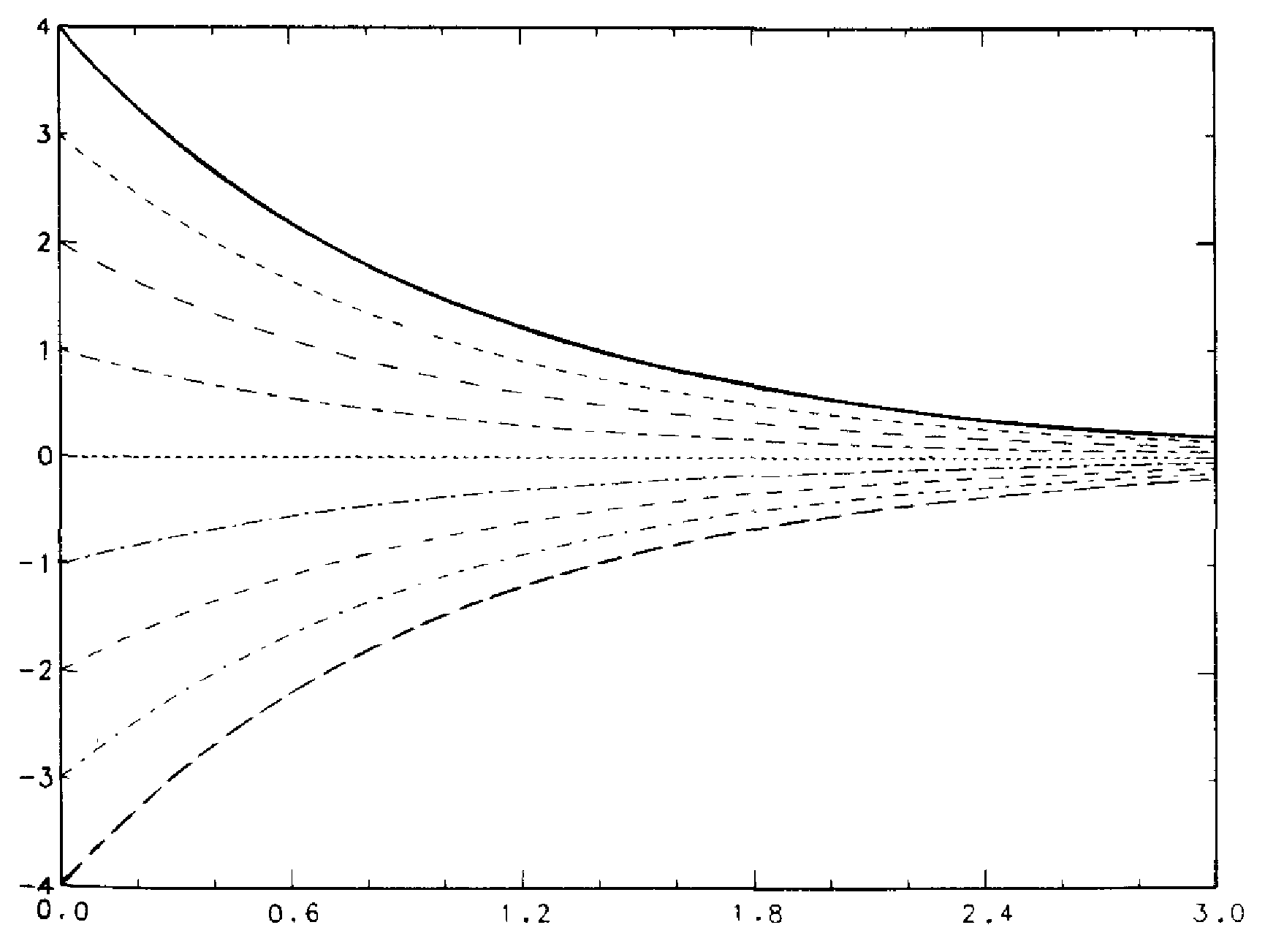
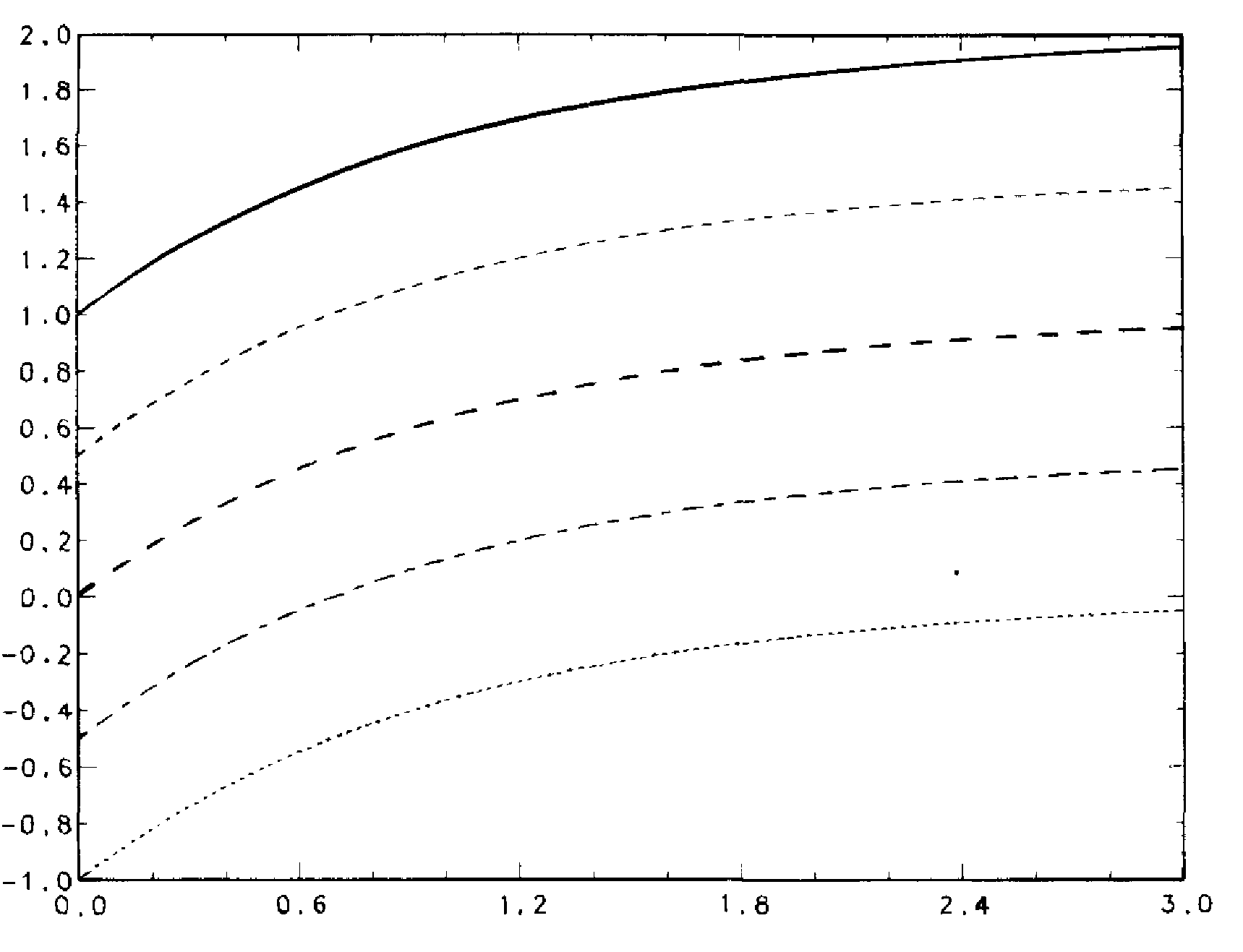


Рис. 8.1. Семейства решений.

Независимая переменная *t* часто имеет смысл времени; при этом можно считать, что ОДУ описывает эволюцию физической системы. Начальное состояние такой системы обычно бывает известным, что и поясняет смысл термина «начальная задача». Если *t*0 и *A* – заданные числа, то начальную задачу записывают в виде



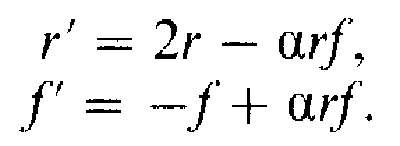
Пара (*t*0, *A*) называется *начальной точкой*.

Если *t* представляет собой время, то решение *y*(*t*) требуется найти на фиксированном интервале *t*0 < *t* < *T*, а если *t* выражает расстояние, то решение можно искать для *T* < *t* < *t*0. И теория, и алгоритмы применимы в обоих случаях, но мы опишем подробно только первый случай.

Различие между семействами решений уравнений *у*' = *f*(*t*) и *у*' = *f*(*t*, *у*) показывает еще одно отличие ДУ от квадратур. При вычислении определенного интеграла, скажем, от 0 до 1 мы вольны вычислять значения подынтегральной функции в узлах квадратурной формулы на [0, 1] в любой последовательности. Действительно, ранее рассмотренный адаптивный квадратурный алгоритм выбирает подынтервалы, исходя из оценки погрешности, а не из положения в исходном интервале интегрирования. Это допустимо в случае квадратур, поскольку подынтегральную функцию всюду можно вычислять непосредственно. Но это невозможно для *y*' = *f*(*t*, *y*): нельзя вычислить значение производной решения в точке t = 0.5 до тех пор, пока в этой точке не будет вычислено решение, так как *y*'(0.5) = *f*(0.5, *y*(0.5)).

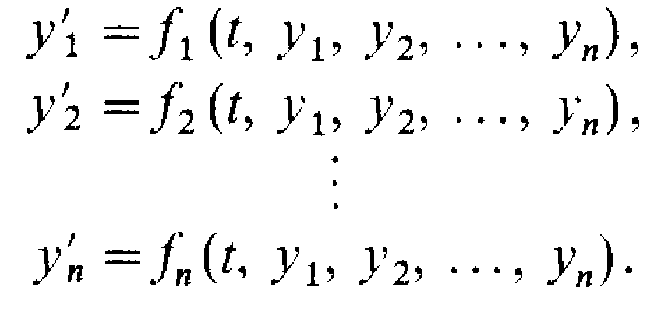
**8.1.2. Уравнения высокого порядка и системы уравнений (326)**

Большинство практических задач приводит не к одному ОДУ первого порядка. Системы уравнений первого порядка могут возникать непосредственно, если в задаче имеется более одного неизвестного. Например, простая модель «хищник – жертва», описывающая популяции кроликов *r*(*t*) и лис *f*(*t*), поедающих только кроликов, записывается в виде системы двух уравнений первого порядка (см. задачу 8.5)



Взаимодействие этих двух популяций пропорционально произведению их численностей с коэффициентом пропорциональности . Задача состоит в том, чтобы определить обе численности *r*(*t*) и *f*(*t*), исходя из их значений, известных в начальный момент времени *t*0.

В более общей ситуации имеется *n* неизвестных функций *yi*(*t*), *i* = 1,..., *n*, для которых выписано *n* ДУ



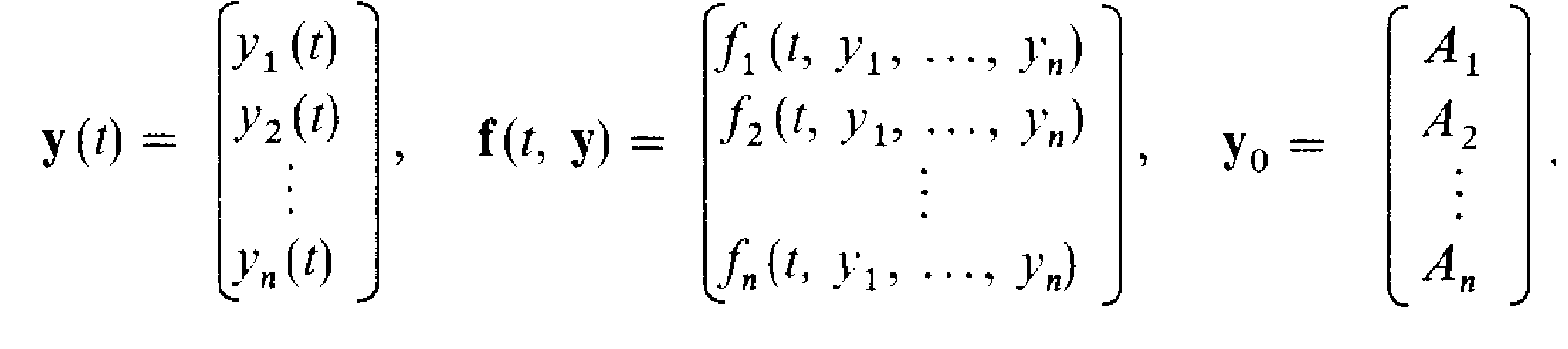
В примере «хищник – жертва» *n* = 2, *y*1(*t*) = *r*(*t*), *y*2(*t*) = *f*(*t*) и



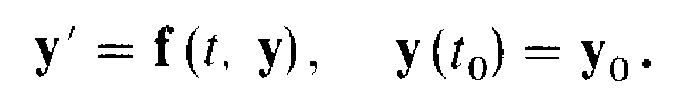
Начальные условия для системы уравнений первого порядка записываются следующим образом:



Стандартный прием – запись системы в векторной форме – позволяет упростить обозначения. Введем



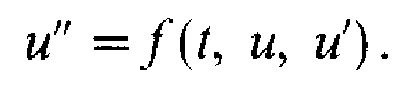
Тогда начальная задача для системы записывается в виде



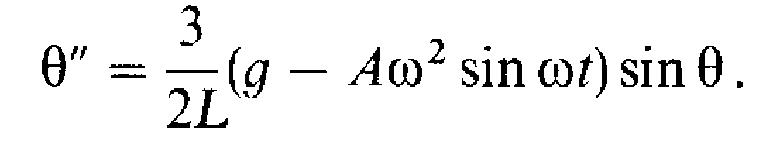
Если начальные условия не заданы, то система обладает *n*-параметрическим семейством решений; каждое решение представляет собой набор из *n* функций.

Поразительно, но теория и численные алгоритмы для одного уравнения в основном переносятся и на случай системы. Задача 8.10 позволит вам убедиться в этом самостоятельно – запрограммировать простой алгоритм и решить систему двух уравнений. Вот почему при изложении большинства последующих вопросов мы будем брать в качестве модели одно уравнение.

ОДУ *второго порядка* – это уравнение вида



Например, перевернутый маятник (см. задачу 8.13) моделируется нелинейным уравнением второго порядка

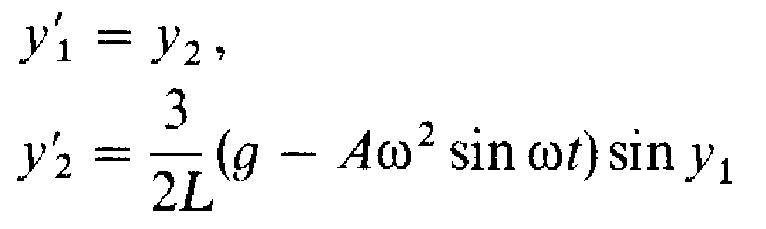


Здесь (*t*)–угол отклонения маятника в момент времени *t*; *A*,  и *L* – физические параметры; *g* – ускорение свободного падения. Начальная задача для этого уравнения содержит также угол  и скорость ', заданные в момент времени *t*0.

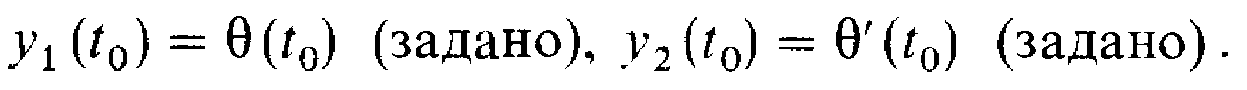
Далее, уравнения высокого порядка обычно преобразуют к системам уравнений первого порядка с помощью стандартной замены неизвестных. Введем две новые неизвестные функции *y*1(*t*) и *y*2(*t*) следующим образом:



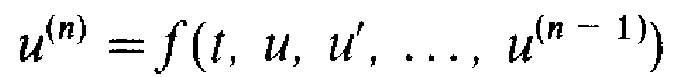
Тогда для новых неизвестных получим систему двух уравнений первого порядка



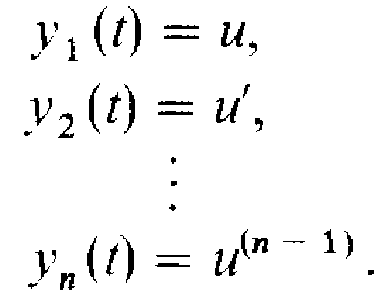
с начальными условиями



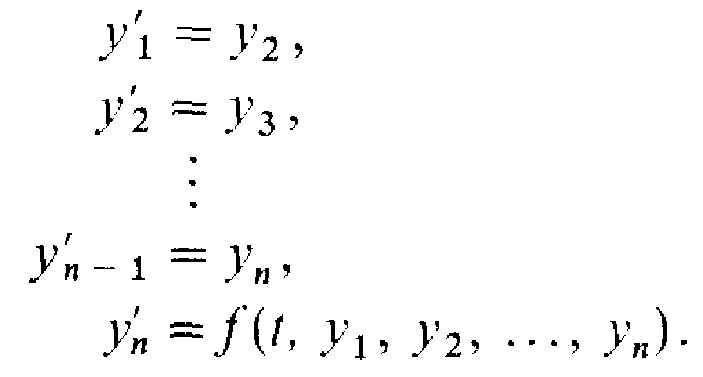
В случае уравнения *n*-го порядка



введем *n* новых неизвестных *y*1(*t*), *y*2(*t*), …,.*yn*(*t*):



Для новых неизвестных система из и уравнений первого порядка принимает вид



Для задачи о маятнике *f*1(*t*, *y*1, *y*2) = *y*2 и *f*2(*t*, *y*1, *y*2) = 3(*g* – *A*2 sin *t*)(sin *y*1)/(2*L*). Отметим, что в системе «лисы – кролики» переменная *t* не входит в правые части *f* явно, а в случае маятника входит (в *f*2). В первом случае системы называют автономными, а во втором – неавтономными. Для наглядности при решении ОДУ второго порядка бывает полезно строить график одной зависимой переменной как функции от другой, оставляя независимой переменной роль параметра. Если зависимыми переменными являются положение и скорость, то такой график называется *фазовой плоскостью*, поскольку положение и скорость именуют двумя *фазами* системы. Кривая, изображающая решение, называется *траекторией* в фазовой плоскости. Это термин широко используется и при решении систем ОДУ произвольного вида; он означает кривую на графике, по осям которого откладываются зависимые переменные. Например, если в качестве зависимых переменных взяты координаты *x* и *y* спутника на плоской орбите (см. задачу 8.6), то фазовой плоскостью будет орбита (рассматриваемая «сверху»); эта картина несет гораздо больше информации, чем два отдельных графика для *x* и *y* как функций от *t*.

Почти все библиотечные программы для начальных задач позволяют решать системы из *n* уравнений первого порядка. Если уравнение пользователя имеет второй или более высокий порядок, то это уравнение следует привести к системе (вручную). Каждый пользователь, желающий работать с наиболее современным программным обеспечением, должен уметь выполнять такие преобразования. Системы, полученные из уравнения *n*-го порядка, гораздо проще систем общего вида, однако лишь немногие из современных программ способны воспользоваться этим преимуществом.

Преобразование уравнения *n*-то порядка к системе из *n* уравнений первого порядка сулит определенную выгоду. Хотя основной интерес представляет искомая функция *y*1(*t*), однако и ее производные, особенно первая производная *y*′(*t*), также физически значимы. Эти производные вычисляются в процессе решения системы «бесплатно».

8.2. **Устойчивые и неустойчивые уравнения и численные методы (329)**

Большинство начальных задач, возникающих на практике, не удается решить аналитически. С XV в. до середины 50-х годов XX в. их решали приближенно с помощью весьма изощренных механических интеграторов. Возможно, этот период оказался столь долгим потому, что результатом работы интеграторов была функция, а не набор чисел. Численные методы восходят к семнадцатому столетию – к Ньютону, который предложил использовать их для расчета траекторий комет. В 1748 году французские ученые предсказали возврат кометы Галлея ценой шести месяцев расчетов вручную «с утра до вечера и даже подчас во время еды». Вычисления вручную продолжали использоваться до 60-х годов XX века. Сегодня наиболее популярная техника численного интегрирования заключается в следующем: вместо того чтобы искать приближенное решение во всех точках *t* отрезка [*t*0, *T*], мы ограничиваемся поиском приближенных значений решения для набора точек *t*0, *t*1, ..., *T*. Если точное решение в точке *tk* равно *y*(*tk*), то приближенное значение обозначим через *yk*. Мы также будем пользоваться обозначением *y*'*k* ≡ *f*(*tk*, *yk*). Отметим, что *y*'*k* не совпадает с *y*'(*tk*). (В разделе 1.2 мы также использовали обозначение *yk* для *k*-й компоненты вектора решения системы. Оба обозначения общеприняты, и их употребление редко приводит к недоразумениям.)

Приближенное значение *yk* вычисляется по найденным ранее величинам *yk*–1, …, *y*1, *y*0. Если формула, по которой вычисляется *y*k, зависит явно только от *yk*–1, то метод называется *одношаговым*. Если *yk* вычисляется по двум значениям *yk*–1 и *yk*–2, то метод называется *двухшаговым*. Многошаговые методы доставляют определенные трудности, например на первом шаге интегрирования.

Набор узлов *tk* может быть задан пользователем или построен автоматически самой программой. В обоих случаях расстояние между узлами, т.е. шаг *hk* ≡ *tk*+1 – *tk*, зависит от конкретной задачи, численного метода и точности, с которой мы хотим вычислить результат *yk*. Мы знаем, что *y*0 = *y*(*t*0) и во многих случаях *y*1 ≈ *y*(*t*1), но *y*1 ≠ *y*(*t*1). Поэтому точка (*t*1, *y*1) уже не будет лежать на кривой решения, исходящей из начальной точки (*t*0, *y*0). Скорее всего, она будет лежать на другой кривой из семейства решений. Если мы проследим такую кривую обратно до *t*0, то не окажемся в точке (*t*0, *y*0). Таким образом, точка (*t*1, *y*1) отвечает решению данного ДУ, но при другом начальном условии. Рис. 8.2 иллюстрирует эту ситуацию.

Лучшее, на что можно надеяться при переходе от *t*1 к *t*2, – это что *y*2 удержится на кривой, проходящей через (*t*1, *y*1). В действительности такое маловероятно. Еще менее вероятно возвращение на исходную кривую, проходящую через начальную точку (*t*0, *y*0). Таким образом, если на первом шаге допущена ошибка, то, даже при отсутствии ошибок в дальнейшем, числа *y*2, *y*3, … также будут неточны. На величины их погрешностей влияют не только численный метод и шаг интегрирования, но и само семейство решений.

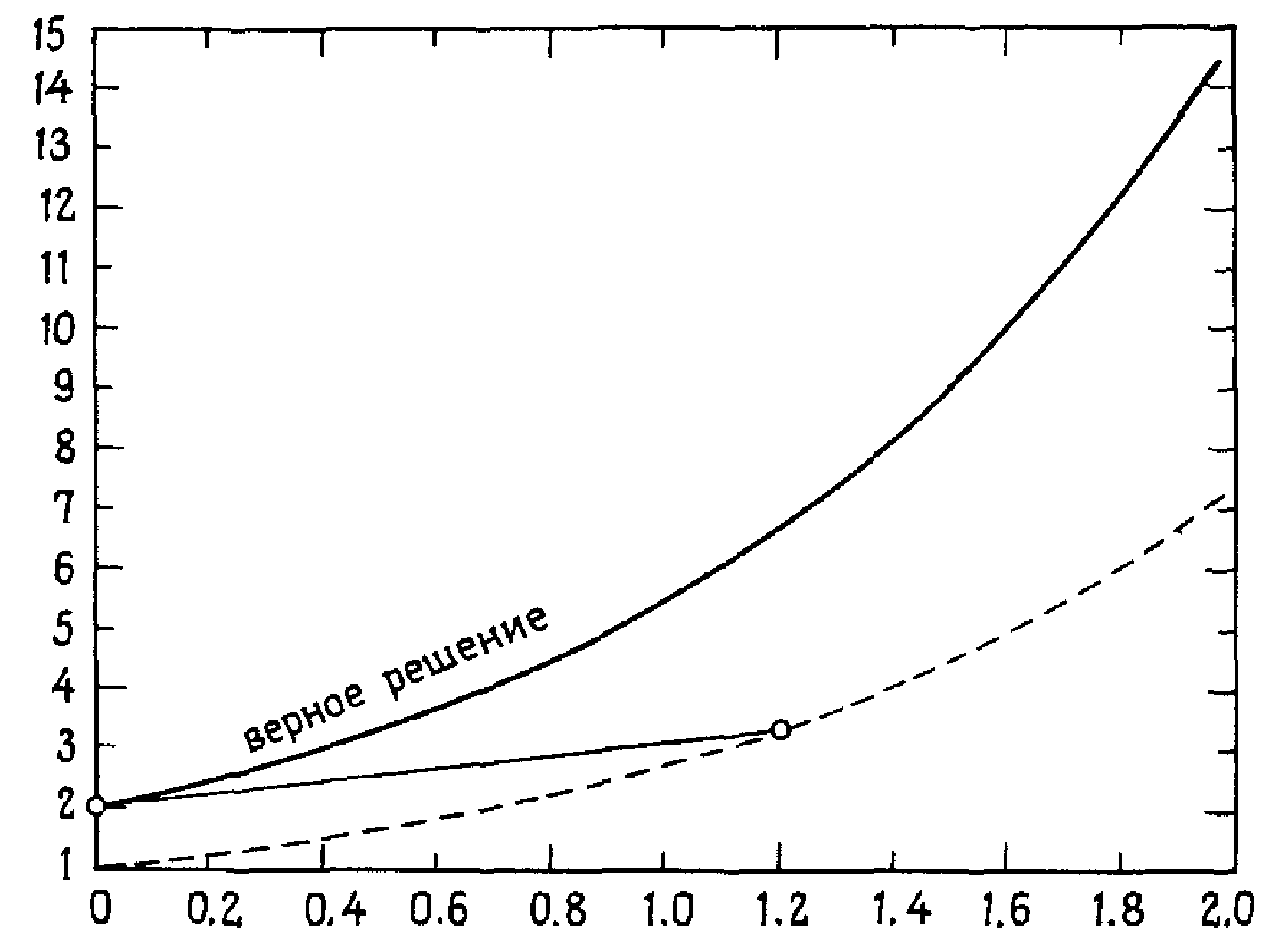


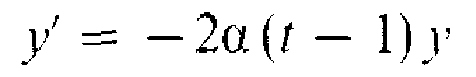
Рис. 8.2. Соскальзывание с кривой решения.

Обратимся к семейству решений *ce*–*t* изображенному на рис. 8.1. Заметим, что кривые из этого семейства сходятся с ростом *t*. Вообще, если кривые решений уравнения *y*' = *f*(*t*, *y*) расходятся по мере удаления *t* от начальной точки, то уравнение называется *неустойчивым*. Если кривые приближаются друг к другу, то говорят, что уравнение *устойчиво*. Уравнение общего вида может демонстрировать на различных частях отрезка интегрирования оба типа поведения. Системы уравнений характеризуются такими же свойствами, хотя их труднее интерпретировать графически. Если уравнение неустойчиво, то ошибки, вообще говоря, имеют тенденцию к росту. Устойчивые уравнения подавляют ошибки. Что происходит на самом деле, определяется не отдельным решением, а всем семейством решений. Две задачи *y*' = *y*, *y*(0) = 1 и *y*' = exp(*t*), *y*(0) = 1 имеют общее решение, но семейства решений соответствующих уравнений различны. Как вы полагаете, какую из этих задач труднее решить численно?

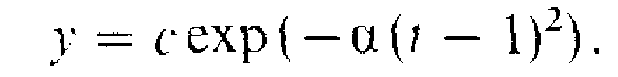
При фиксированном *t* угол наклона касательных к кривым из семейства решений, вообще говоря, изменяется как функция от *y*. Это изменение определяется величиной *fy*, которая служит мерой устойчивости уравнения. Число *fy* называется якобианом уравнения и обозначается через *J*. Устойчивым уравнениям соответствуют отрицательные значения *J*. Пример тому вы можете увидеть на рис. 8.1. В заключение скажем, что не следует рассчитывать на высокую точность при численном решении неустойчивого ОДУ

Пример 8.2. **Устойчивое и неустойчивое уравнения**. (331)

Уравнение



обладает семейством решений



Некоторые кривые из этого семейства для случая  = 5 и интервала [0, 2] изображены на рис. 8.3. Мы видим, что эти кривые вначале удаляются друг от друга, а затем сближаются. Это согласуется с тем фактом, что якобиан  *J* = –2(*t* – 1) положителен при *t* < 1 и отрицателен при *t* > 1. Это области умеренной неустойчивости и устойчивости соответственно. Изменение начального условия при *t* = 0 с 0.01 до 0.1 приводит к росту решения в точке *t* = 1 с 1.0 примерно до 15.0.

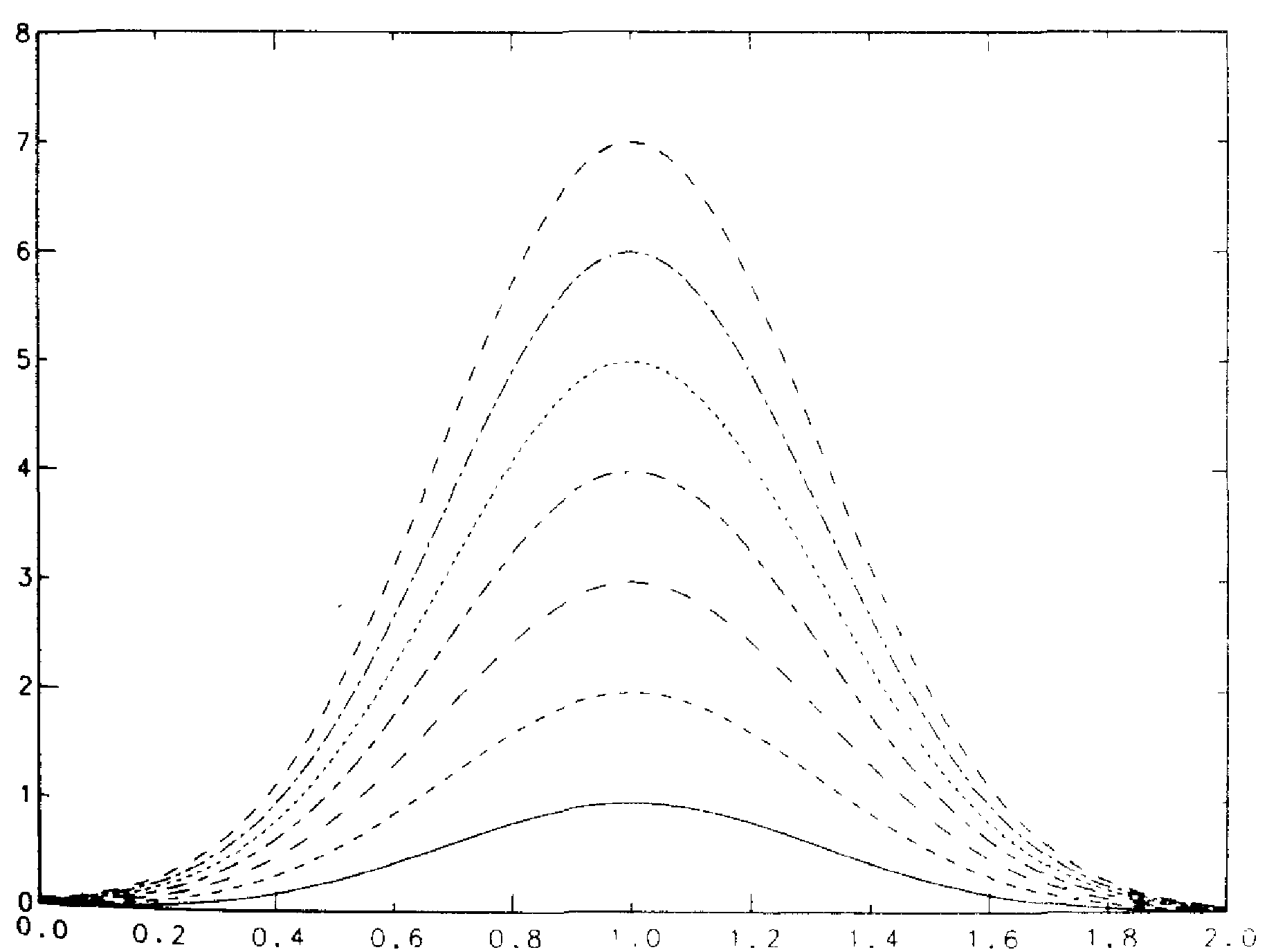


Рис. 8.3. Уравнение с областями устойчивости и неустойчивости.

8.3. **Жесткие** ДУ (334)

ОДУ называют *сверхустойчивым*, если *J* « 0. Мы вправе рассчитывать, что численное решение начальных задач для таких уравнений окажется значительно проще, чем для неустойчивых уравнений, поскольку здесь соскальзывание с правильной кривой решения не будет приводить к чересчур разрушительным последствиям. Мы увидим, однако, что это будет зависеть от численного метода. Обычно сверхустойчивые уравнения называют жесткими, хотя этот термин больше подходит для систем уравнений, а не для одного уравнения. Задачи, возникающие на практике, обычно бывают устойчивыми, жесткими или лишь умеренно неустойчивыми.

Жесткость – это довольно сложное понятие, которому мы не дали строгого определения. Рабочее описание жесткой задачи таково: это задача, моделирующая физический процесс, компоненты которого обладают несоизмеримыми характерными временами, или же процесс, характерное время которого намного меньше отрезка интегрирования. (Более удовлетворительное описание, указывающее на роль величины собственных значений якобиана *J*, приведено в разделе 2.1.) Многие практики отождествляют жесткость и «трудность», однако это неверно, поскольку, чем жестче система, тем лучше ее решает хорошая «жесткая» программа. На самом деле этот термин следует связывать со всем комплексом, включающим задачу, отрезок интегрирования, применяемый численный метод и даже малозначительные на поверхностный взгляд детали реализации алгоритма. Читатель, желающий ознакомиться с этой темой глубже, может найти много полезных вводных статей и ссылок в книге [Aiken, 1985] или в обзорной статье [Byrne, Hindvarsh, 1987]. Жесткие задачи возникают почти во всех прикладных областях, но особенно характерны для химической кинетики.

**Пример 8.3. Жесткое уравнение.(334)**

Рассмотрим начальную задачу *y*' = –(*y* – sin *t*) + cos *t*, *y*(0) = 1. Ее решение имеет вид



Если мы хотим решить эту задачу на отрезке 0 < *t* < 1 и  велико, например a ≈ 1000, то *J* = –1000 и задача является жесткой. (На ином отрезке интегрирования, скажем при *t* < 0.002, эта задача не будет считаться жесткой.) Для *t*, близких к нулю, решение *y* быстро уменьшается и падает почти до 0. Потом оно практически не отличается от sin *t*. Решение вблизи *t* = 0 качественно отличается от решения при больших *t*. Начальный отрезок решения называется *пограничным слоем*, остальная его часть характеризуется *режимом медленного* (*плавного*) изменения. На рис. 8.4 проиллюстрировано семейство решений этой задачи. Важно отметить, что всякое решение, исходящее из какой-либо точки плоскости, расположенной выше или ниже плавно меняющейся кривой sin *t*, очень быстро стремится к этой кривой. На приведенном рисунке для лучшей визуализации графиков мы заменили константу 1000 на 20; у новой задачи жесткость гораздо меньше.

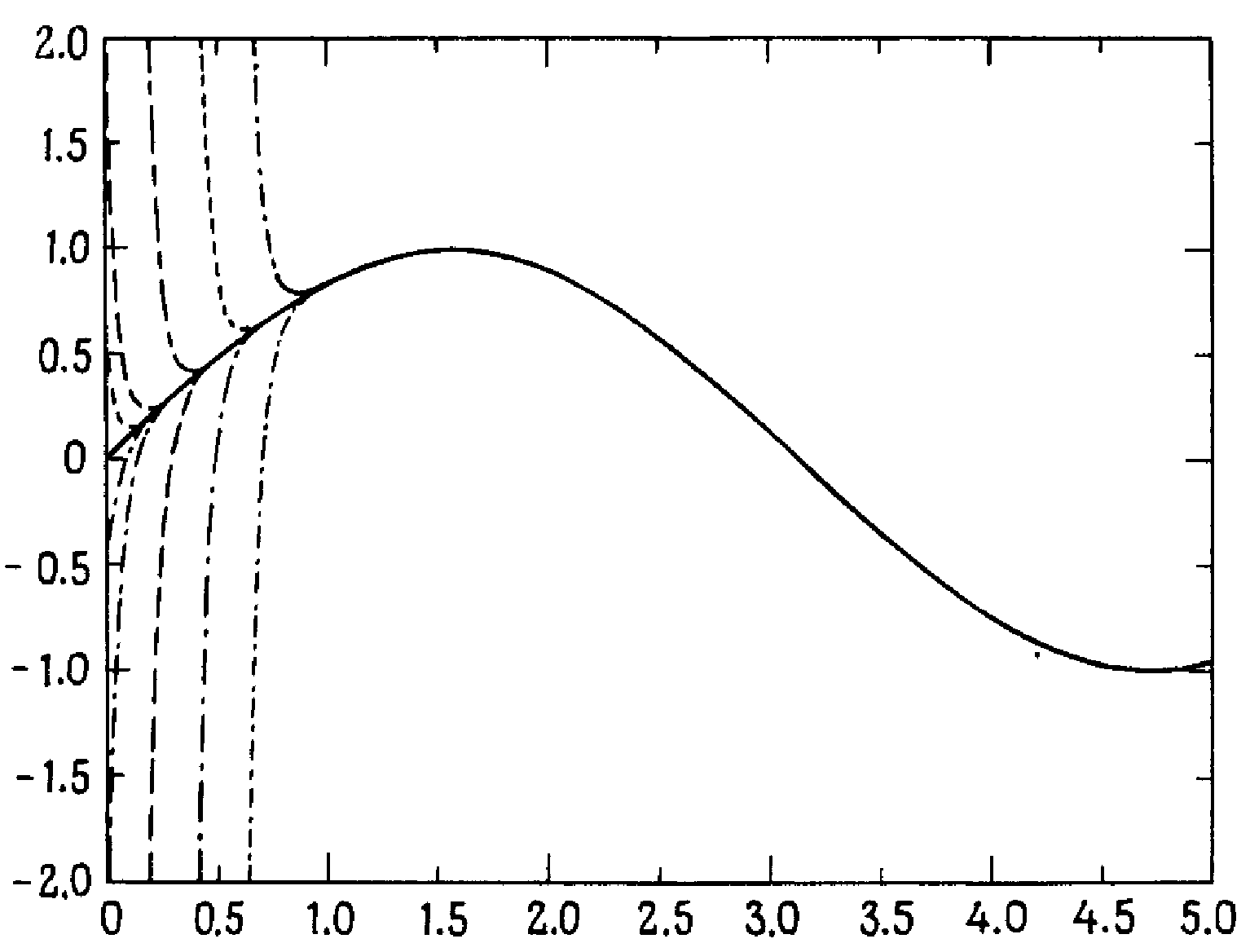


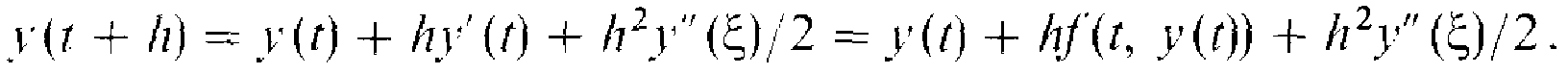
Рис. 8.4. Жесткое уравнение.

На практике жесткие уравнения часто характеризуются обоими типами поведения, поскольку моделируют физические процессы, протекающие с различными характерными временами. Часто пограничный слой возникает не в начале интегрирования, а лишь тогда, когда некоторый «управляющий» параметр резко изменит свое значение. Позже мы увидим пример такого поведения, когда будем рассматривать модель количества озона в атмосфере. В некоторых приложениях важен пограничный слой; в других – лишь предельное поведение (плавное изменение) при больших временах интегрирования. Об этом различии важно помнить, продумывая способ решения задачи. Можно ожидать, что отследить пограничный слой окажется довольно трудно, а медленное изменение – легче. Мы увидим, что при численном решении такого рода задач наши ожидания подтверждаются не всегда, если только мы не выберем численный метод очень тщательно.

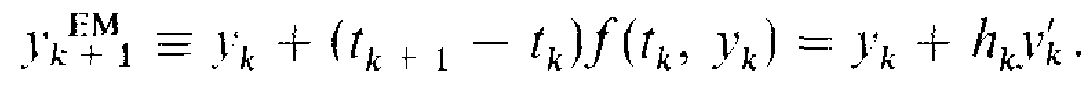
8.4. **Метод Эйлера (336)**

Теорема Тейлора служит основным инструментом вывода большинства численных методов решения ОДУ. Конечно, цель настоящей главы не в том, чтобы конструировать методы, а в том, чтобы помочь читателю понять, как устроена и как работает некоторая конкретная программа. Однако ради этого нам придется вывести несколько простых методов, в том числе метод Эйлера, обратный метод Эйлера и метод трапеций. Мы сделаем это, чтобы проиллюстрировать такие понятия, как порядок, точность и устойчивость.

Предположим, что требуется решить задачу *y*' = *f*(*t*, *y*), *y*(*t*0) = y0 на интервале [*t*0, *T*]. Допустим, что решение *y*(*t*) имеет по крайней мере непрерывную вторую производную. Тогда



Если известно приближенное решение *yk* в узле *tk* и выбран новый узел *tk* + 1, то для *метода Эйлера* – одношагового метода – значение *y*EM*k*+1 (где индекс EM означает метод Эйлера – the Euler Method) определяется по выписанной выше формуле, в которой отброшен остаточный член:



Каханер Д., Моулер К., Нэш С. Численные методы и математическое обеспечение: – М.: Мир, 1998.–575 с.