Решение дифференциальных уравнений

Если при решении аналитических уравнений и систем мы искали *значения* (скалярные или векторные), превращающие уравнения в тождества (искали *корни* системы уравнений), то решение *дифференциальных уравнений* в общем случае сводится к поиску *функций*, подстановка которых и их производных разного порядка также превращает исходное уравнение или систему уравнений в тождества. В частном случае такое решение сводится к нахождению не самих функций, а их значений в определенной точке, либо таблицы значений искомой функции (искомых функций).

Здесь, можно отметить некую тенденцию, которую одни считают "нехорошей", а другие — прогрессивной[1] или, по крайней мере, неизбежной — отход от аналитических методов решения задач с разбиением их на особые классы (линейные, нелинейные, алгебраические, степенные и т. д.) и повсеместный переход к численным методам с реализацией на компьютере.

> ***Примечание***
> Здесь слово "аналитических" не тождественно слову "символьных", за которым стоят компьютерные инструменты аналитического решения задач — символьная математика Mathcad или Maple/MuPAD[2], в частности. Хотя эти инструменты применяются и для решения дифференциальных уравнений.

"Прогрессисты", если так можно выразиться, утверждают, что курс дифференциальных уравнений (да и вся высшая математика) в технических вузах читается по учебникам XVIII века. То есть учебники-то новые, но содержание в них старое, не менявшееся уже триста лет и не учитывающее современные средства решения математических задач. Это в частности приводит к неоправданному утяжелению и усложнению многих технических дисциплин. Конкретный пример. Автор в студенческие годы слушал курс теории автоматического регулирования и ничего в нем толком не понимал, хотя интуитивно чувствовал, что курс очень "красивый" с точки зрения математики. Сначала автор объяснял такое свое недопонимание тем, что преподаватель якобы был не очень хорош[3]. Но потом автор понял, что дело было в другом. Этот курс в таком виде, в каком его читали 30—40 лет назад и читают до сих пор, состоял на 20% собственно из самого курса и 80% из материала, прямо не относящегося к курсу, а связанного с тем, как, допустим, дифференциальные уравнения, получающиеся при постановке задачи автоматического регулирования, можно решить не в лоб, а через различного рода ухищрения — линеаризацию, прямое и обратное интегральные преобразования, частотно-амплитудные и прочие характеристики и т. д. Представьте себе, что на лекции по математике (а конкретнее — теории чисел) преподаватель станет дополнительно объяснять, как столбиком перемножают

---

[1] Термины "хороший" и "прогрессивный" не всегда являются синонимами.

[2] Разработчика пакета MuPAD купила фирма — разработчик пакета Matlab. Поэтому фирма PTC в будущем сменит символьное ядро пакета Mathcad.

[3] А студенты, как известно, очень редко винят именно себя в плохом знании предмета.

два числа, вольно или невольно умалчивая при этом, что есть такое достижение цивилизации как калькулятор или компьютер. Подобными побочными отступлениями можно "убить" любую даже самую занимательную дисциплину и перекрыть доступ к ее "вычислительной части", когда можно на компьютере быстро и достаточно точно решить дифференциальное сложное уравнение, а затем "поиграть" исходными данными и получить новый ответ в виде графиков или даже анимационных клипов.

Этот феномен (преподавание математики и, в частности, дифференциального исчисления в технических вузах по учебникам XVIII века) автор совместно с А. П. Солодовым[4] попытался раскрыть в книге "Mathcad. Дифференциальные модели"[5] ("западный" вариант — "Differential Models. An Introduction with Mathcad". Springer. 2004 — см. **www.thermal.ru**).

Вспомним, как сейчас в вузах читается курс по дифференциальным уравнениям?! Преподавателем дается дифференциальное уравнение, а откуда оно взялось, как оно решается современными инструментами, как можно визуализировать полученные решения средствами двух- и трехмерной графики или анимацией — об этом часто опять же вольно или невольно умалчивается[6]. Все эти компьютерные новшества считаются своего рода математической "попсой", которой не место на "классических" лекциях по математике[7]. Да, выпускнику физмата МГУ достаточно мельком взглянуть на аналитическое решение дифференциального уравнения или даже на само уравнение, чтобы увидеть не только трехмерную, но и n-мерную ($n > 3$) графику решения или решение в неком движении (анимацию). Рядовой же инженер или студент может все это увидеть только на экране компьютера при соответствующей постановке задачи и то далеко не всегда — проблема "смотреть, но не увидеть".

У автора в школьные годы на уроках арифметики был устный счет, состоявший в том, что учитель давал задание ученику перемножить 84 на 95, например, а сам считал до десяти. Если ответа не было или он был неверный, то в журнал ставилась двойка. Считать же учили "не в лоб", а с применением различных "вычислительных хитростей": "84 множим на 100 и отнимаем 840/2 и т. д.". При этом учитель подчеркивал такую мотивацию: "Пойдешь в магазин, а там тебя обсчитают и обвесят, если

---

[4] А его можно считать соавтором данной главы.

[5] Солодов А. П., Очков В. Ф. Mathcad. Дифференциальные модели. — М.: Издательство МЭИ, 2002. Еще один соавтор этой главы книги — В. И. Коробов. См. книгу Коробов В. И., Очков В. Ф. Химическая кинетика: введение с Mathcad, MA/CS и Maple. — М.: Издательство Горячая линия-Телеком, 2008.

[6] Приятное исключение — книга Плис А. И., Сливиной Н. А. Mathcad. Математический практикум для инженеров и экономистов. — М.: Финансы и статистика, 2003. В ней излагаются азы высшей математики с иллюстрациями в среде Mathcad.

[7] Когда автор предложил одному своему коллеге-математику читать лекции в современной аудитории с компьютером, Интернетом, экраном, мультимедийным проектором и иллюстрировать лекционный материал "живыми" решениями задач в средах каких-либо математических программ, то он полушутя-полусерьезно ответил, что можно еще привлечь и "пританцовку", какая сопровождает певцов на эстраде. Доказал теорему — и тут на сцену, пардон, в аудиторию с визгом вбегают полуголые девицы и лихо отплясывают канкан... Можно лекцию читать также и под фонограмму, а под потолком аудитории устроить лазерное шоу — показывать, например, поверхность решения дифференциального уравнения в частных производных...

ты слаб в устном счете!". Теперь такая мотивация не работает — у каждого школьника есть калькулятор, которым, правда, часто запрещают пользоваться на уроках математики[8], да и в современном магазине с электронными весами он не поможет — сейчас там практикуют другие методы "обсчета и обвеса". Но эта мотивация была не главная. В те времена преподаватели даже в периферийных школах[9] были сплошь с университетским образованием, и для них смысл устного счета состоял в том, что математика — это лучшая *гимнастика для ума*. Калькулятор же здесь может быть полезен примерно так же, как гидроусилитель у спортивного тренажера. Сейчас появились "суперкалькуляторы", умеющие не только складывать и умножать, но и брать производные и интегралы, искать пределы, решать дифференциальные уравнения и т. д. И вновь встает вопрос о запрещении таких устройств на уроках уже не арифметики, а высшей математики, если опять же рассматривать этот предмет как гимнастику для ума, а не только как средство решения практических задач параллельных курсов (физика, химия, механика, термодинамика, сопротивление материалов и т. д.). Автор нисколько не умаляет важность знания "вычислительных хитростей" при решении математических задач. Он только обращает внимание на то, что эти знания должны приобретаться параллельно с приобретением знаний и навыков от компьютеров для решения подобных задач.

А теперь рассмотрим инструменты решения дифференциальных уравнений в среде Mathcad на нескольких довольно простых, но, автор надеется, занимательных примерах. Главное здесь — раскрыть особенности Mathcad для решения подобных задач.

1. Эпидемия

В городе с населением 20 000 человек (рис. 1) появляются 50 инфекционных больных, что вызывает эпидемию. Предположим, что прирост больных за день пропорционален произведению числа здоровых (еще не переболевших и не приобретших иммунитет) на число больных. Коэффициент пропорциональности `Пр` интегрирует разного рода меры профилактики. Если, к примеру, жители города будут носить марлевые повязки или сделают прививки, то этот коэффициент уменьшится. Спрашивается, как развивается эпидемия, как изо дня в день меняется число больных[10].

---

[8] На уроках арифметики прошлого, "безкалькуляторного" века.

[9] Это была школа № 2 г. Перово-Поле Московской области, с четырьмя начальными классами в трех комнатах "с коридором", где учились в три смены.

[10] Одно из основных потребительских качеств компьютера — это отношение цены к производительности. С подобным критерием можно подойти и к примерам, входящим в пакеты программ. Только вместо производительности тут нужно рассматривать занимательность задачи, а вместо цены — размер соответствующего файла. В этом смысле рекордсмен пакета Mathcad — задача "Эпидемия". Ее уникальность еще и в том, что она — единственная, автор которой (John Truxal) был отмечен в разделе Acknowledgments документации пакета Mathcad. Автор данной книги развил решение этой задачи, показав на ее примере инструменты решения уже не разностных, а дифференциальных схем (уравнений).

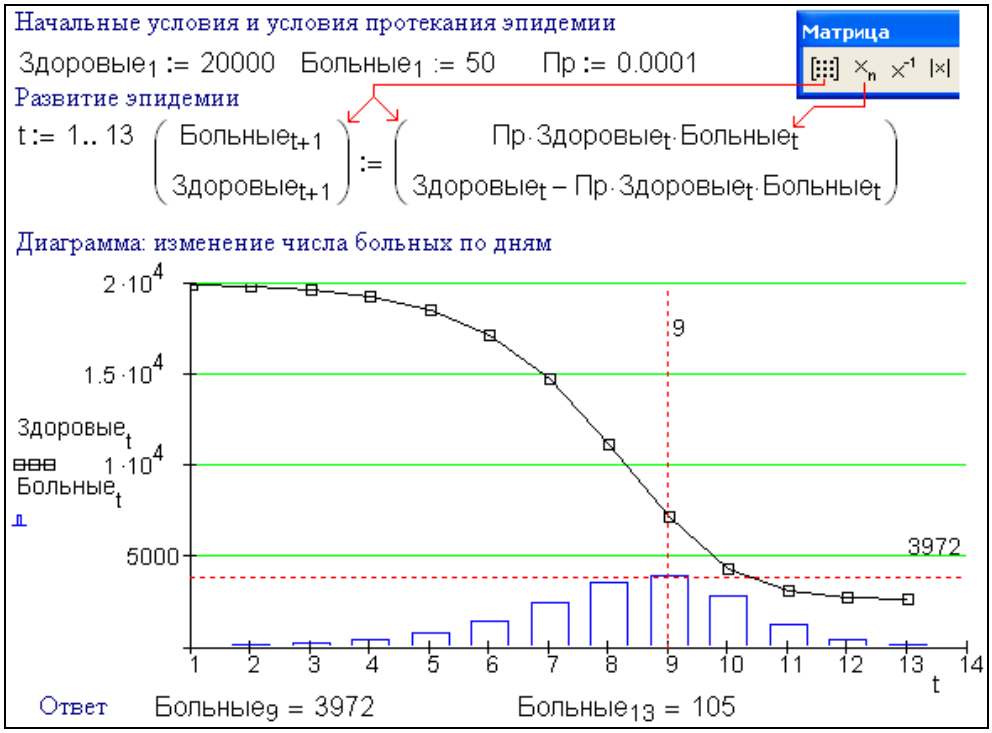


Рис. 1. Задача об эпидемии — разностная схема

Все это словесное описание модели эпидемии легко вмещается в Mathcad-документ (рис. 1) с двумя формулами, объединенными в вектор, что эквивалентно такой BASIC-конструкции:

```
For t = 1 To 13
  Больные(t + 1) = Пр * Больные(t) * Здоровые(t)
  Здоровые(t + 1) = Здоровые(t) – Больные(t)
Next
```

Если бы два выражения на рис. 1 не были заключены в "векторные" скобки, то их выполнение сразу бы прерывалось сообщением об ошибке. Система Mathcad пыталась бы сначала полностью заполнить вектор `Больные`, а уже потом — вектор `Здоровые`. Скобки изменяют порядок счета: он ведется не по строкам, а по столбцам: сначала заполняются вторые элементы векторов `Больные` и `Здоровые` (первые элементы заполняются в начальных условиях)[11], а потом третьи и т. д. Скобки меняют естественный порядок выполнения операторов — они выполняются не слева направо и не сверху вниз, а как бы крест-накрест. Результаты расчета графически отображены на рис. 1.

[11] Там еще есть и пустые нулевые элементы, т. к. мы забыли написать в начале документа `ORIGIN:=0`. Вернее, намеренно не написали, чтобы не утяжелять расчет.

Примечание

Здесь используются два типа декартова графика: линия с точками в виде квадратиков и так называемая bar-diagram — плоская столбчатая диаграмма. Гладкие линии здесь не совсем уместны, т. к. значения меняются дискретно по дням, а не непрерывно.

На тринадцатый день (спад эпидемии) в городе было 105 больных. Критическая точка — девятый день (3972 больных), ради поиска которой и затевают весь этот "расчетный сыр-бор": *моделируя* эпидемию, мы можем планировать работу санитарных служб города — подвезти в аптеки лекарства, отозвать врачей из отпуска, выписать из больниц выздоравливающих и т. д. Вспомним, что самые мощные компьютеры (суперкомпьютеры) этим только и заняты — на них моделируются различные процессы реальной жизни — изменение климата на нашей планете, например.

2. Дифференциальные уравнения на примере эпидемии

Описанная задача об эпидемии сводится к решению *задачи Коши* для системы двух *обыкновенных дифференциальных уравнений* первого порядка относительно двух неизвестных функций `Больные(t)` и `Здоровые(t)`.

$$Больные'(t) = Больные(t) \cdot (Пр \cdot Здоровые(t) - 1)$$

$$Здоровые'(t) = -Пр \cdot Больные(t) \cdot Здоровые(t)$$

На рис. 1 по сути реализован *метод Эйлера*[12] с единичным шагом интегрирования Δt.

$$Больные_{(t+1)} = Больные_t + \Delta t \cdot Больные'_t$$

$$Здоровые_{(t+1)} = Здоровые_t + \Delta t \cdot Здоровые'_t$$

В среде Mathcad до версий PLUS 5.0 дифференциальные уравнения без особых ухищрений можно было решать только методом Эйлера, который имеет низкую точность и производительность (плата за простоту). Инструментарий для решения дифференциальных уравнений (систем) различного порядка и различными методами появился в арсенале Mathcad PLUS 6.0. В него входили 13 встроенных функций (`Bustoer`, `bustoer`, `bvalfit`, `multigird`, `relax`, `Rkadapt`, `rkadapt`, `rkfixed`, `sbval`, `Stiffb`, `stiffb`, `Stiffr` и `stiffr`), работа одной из которых (самой, наверно, востребованной `rkfixed` — метод Рунге — Кутта *(rk)* четвертого порядка с фиксированным *(fixed)* шагом интегрирования) показана на рис. 2. У этой функции пять аргументов:

- вектор (скаляр при одном уравнении) начальных значений искомых решений (задача Коши);
- абсцисса начальной точки интегрирования;
- абсцисса конечной точки интегрирования;

[12] Или Ойлера — что в него заложено, можно увидеть на сайте **http://twt.mpei.ac.ru/mas/worksheets/Euler.mcd**.

- число шагов интегрирования;
- функция-вектор правых частей системы.

> ***Примечание***
> Обычно тут пишут X_0 и X_1 (два элемента одного вектора), но мы написали хитрее и понятнее: $Люди_{Больные}$ и $Люди_{Здоровые}$, предварительно определив, что индекс Больные равен нулю, а индекс Здоровые — единице (см. для сравнения рис. 12, где обошлись без этой хитрости).

Функция rkfixed возвращает в матрицу (у нас она имеет имя M) с P+1 столбцами и n строками (P — количество уравнений или порядок уравнения — у нас P=2) таблицу решений системы: первый (вернее, нулевой) столбец — это значения аргумента t (их задает пользователь через величины $t_{нач}$, $t_{кон}$ и n), а последующие столбцы — значения ординат решения.

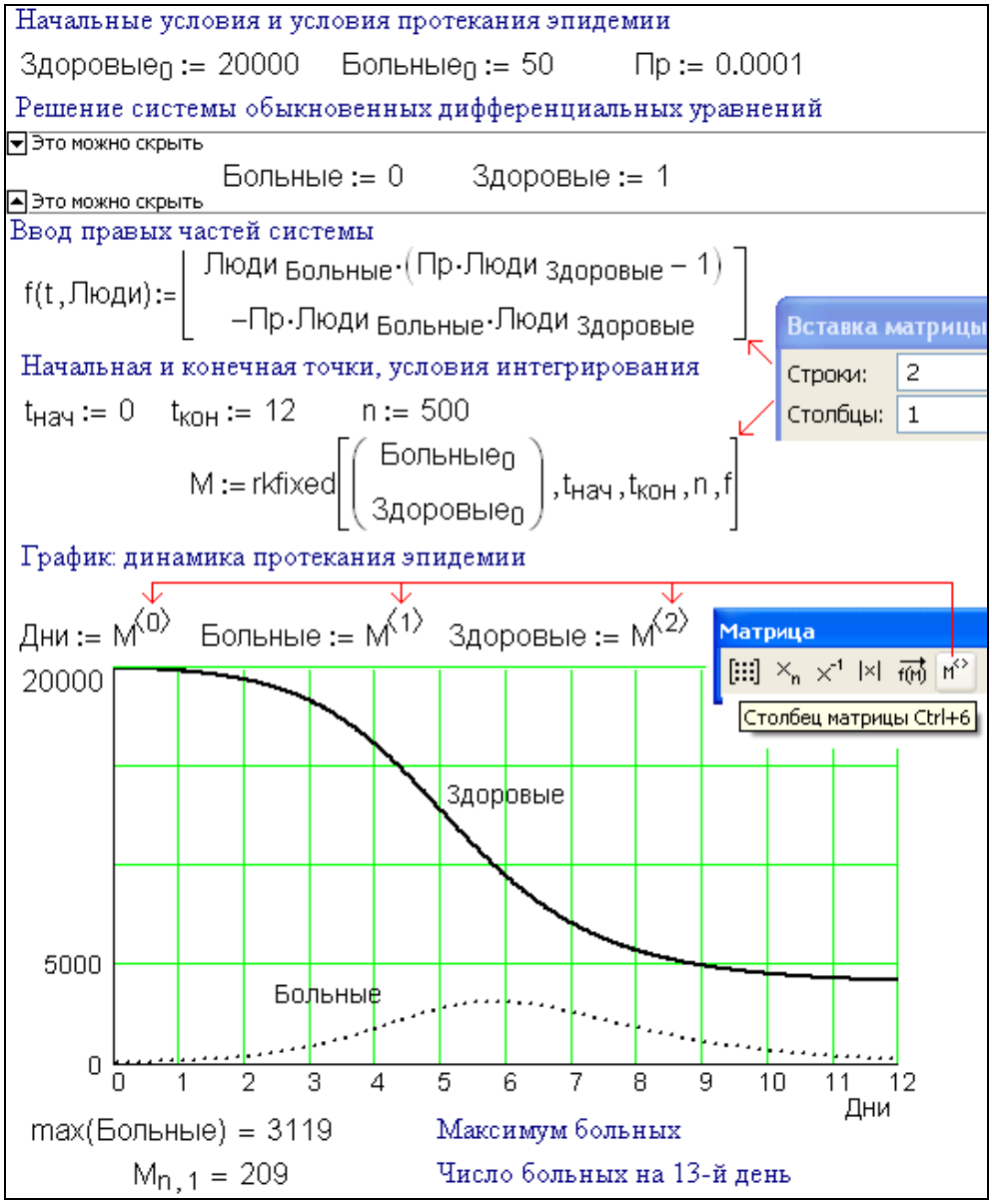


Рис. 2. Задача об эпидемии — решение системы дифференциальных уравнений с помощью функции `rkfixed`

На рис. 2 три столбца матрицы `M` (время, число больных и число здоровых) разнесены по отдельным векторам и отображены графически, что позволяет, как и на рис. 1, проследить динамику развития эпидемии.

На рис. 2 получены несколько иные результаты, чем на рис. 1, хотя характер кривых сохранился. Это объясняется различными значениями точности расчетов (на рис. 1 делалось 13 шагов интегрирования, а на рис 5.2 — 500 шагов) и различными применеными методиками (Эйлер против Рунге и Кутта[13]). На рис. 3 отображена страница Интернета с адресом **http://twt.mpei.ac.ru/mas/worksheets/Euler.mcd**, где можно провести сравнение этих двух методов (Euler и Runge — Kutta).

[13] Многие математики полагают, что есть только один метод Эйлера. Все остальное — это "жалкие" модификации этого метода. Второе категоричное утверждение: есть только один метод решения аналитических уравнений и систем — метод Ньютона *(см. главу 2 и рис. 3.35)*. Все остальное — это опять же "жалкие" модификации...

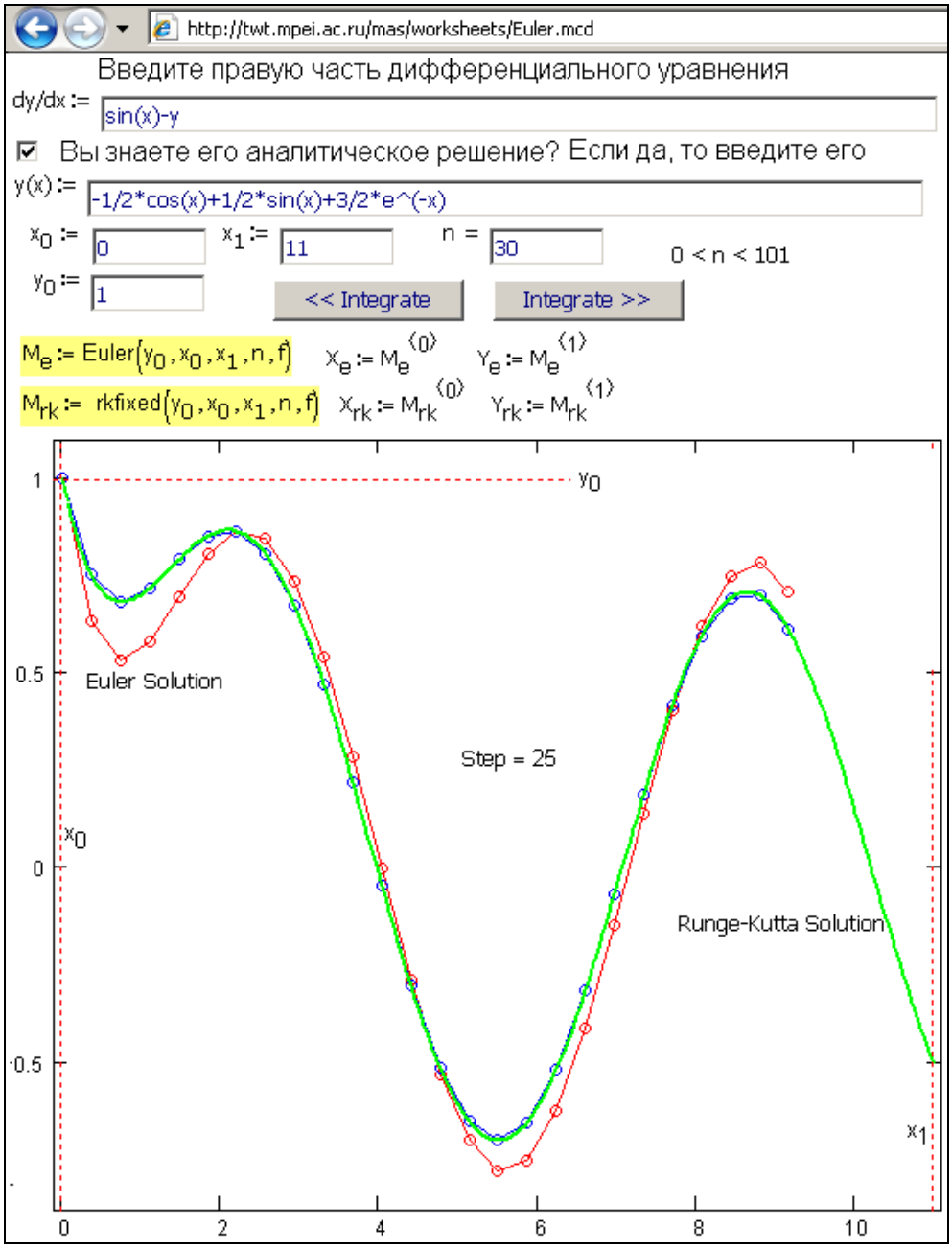


Рис. 3. Методы Эйлера и Рунге — Кутта с фиксированным шагом в Интернете

Из рис. 3 видно, что при малом числе шагов интегрирование (n=30) "Эйлера заносит на поворотах". Исправить эту ошибку можно через увеличение n. Но не всегда. Во-первых, это может недопустимо замедлить расчет, а во-вторых, есть класс задач, где такое механическое увеличение приводит к срыву решения задачи. На отмеченном сайте можно увидеть и Mathcad-программы, раскрывающие методы Эйлера и Рунге — Кутта четвертого порядка.

Можно решить задачу более точно и более быстро, если уменьшать шаг (у нас это Δt) там, где производная меняется быстро, и увеличивать шаг там, где она ведет себя более спокойно. Для этого предусмотрена функция Rkadapt (от англ. *adaptation* — адаптация). Но, несмотря на то, что при решении дифференциального уравнения функция Rkadapt использует непостоянный шаг, она, тем не менее, представит ответ для точек, находящихся на одинаковом расстоянии, заданном пользователем. Аргументы и матрица, возвращаемая функцией Rkadapt, такие же, как при rkfixed. На рис. 4 отображена еще одна страница Интернета с адресом **http://twt.mpei.ac.ru/mas/worksheets/rkadapt.mcd**, где читатель может проследить "динамику" решения дифференциального уравнения при переменном (подстраивающемся) шаге интегрирования. Там же можно увидеть и соответствующую Mathcad-программу метода Rkadapt.

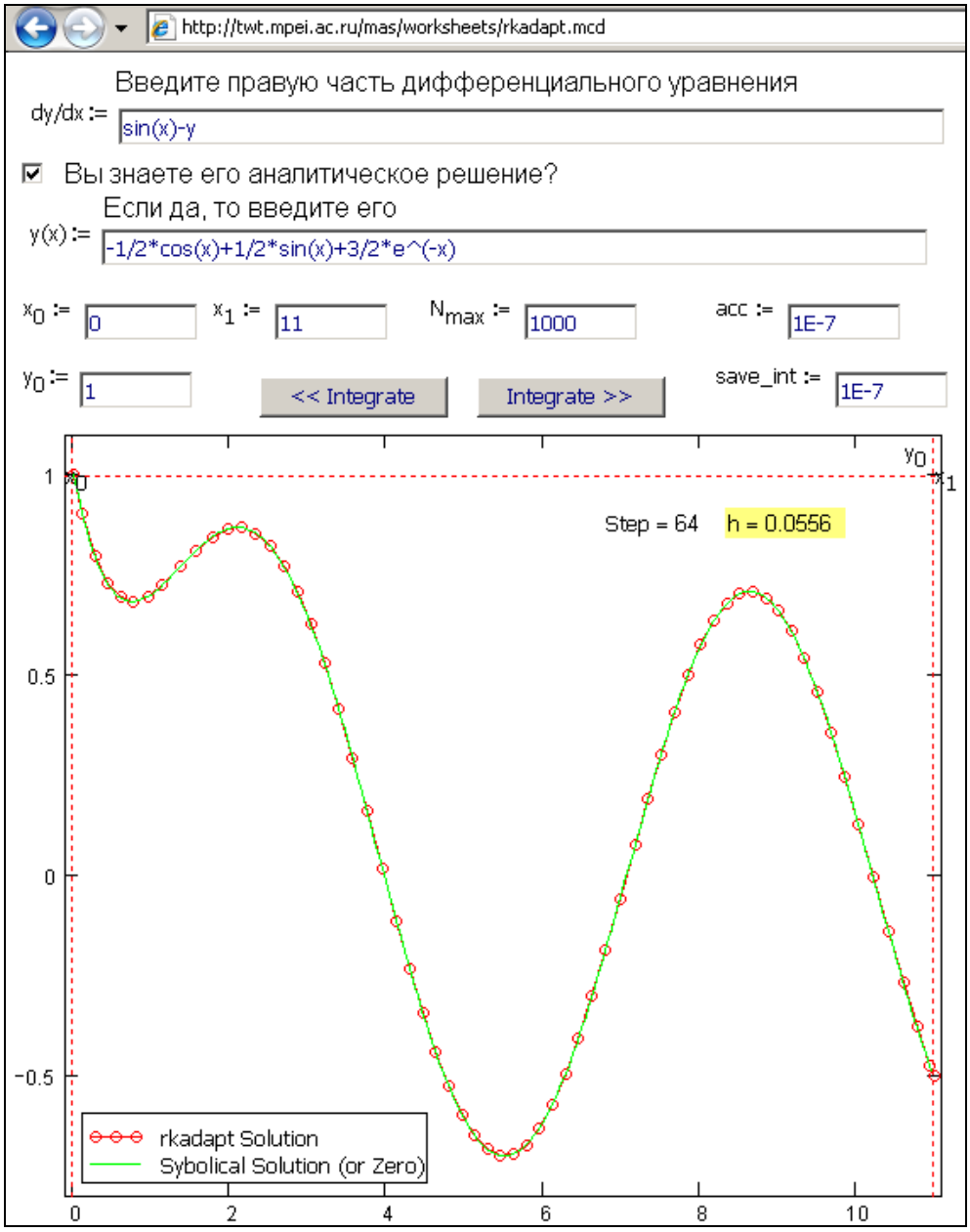


Рис. 4. Метод Рунге — Кутта с переменным шагом в Интернете

На рис. 5 показано сравнение методов Эйлера и Рунге — Кутта четвертого порядка на решении нашей задачи о развитии эпидемии.

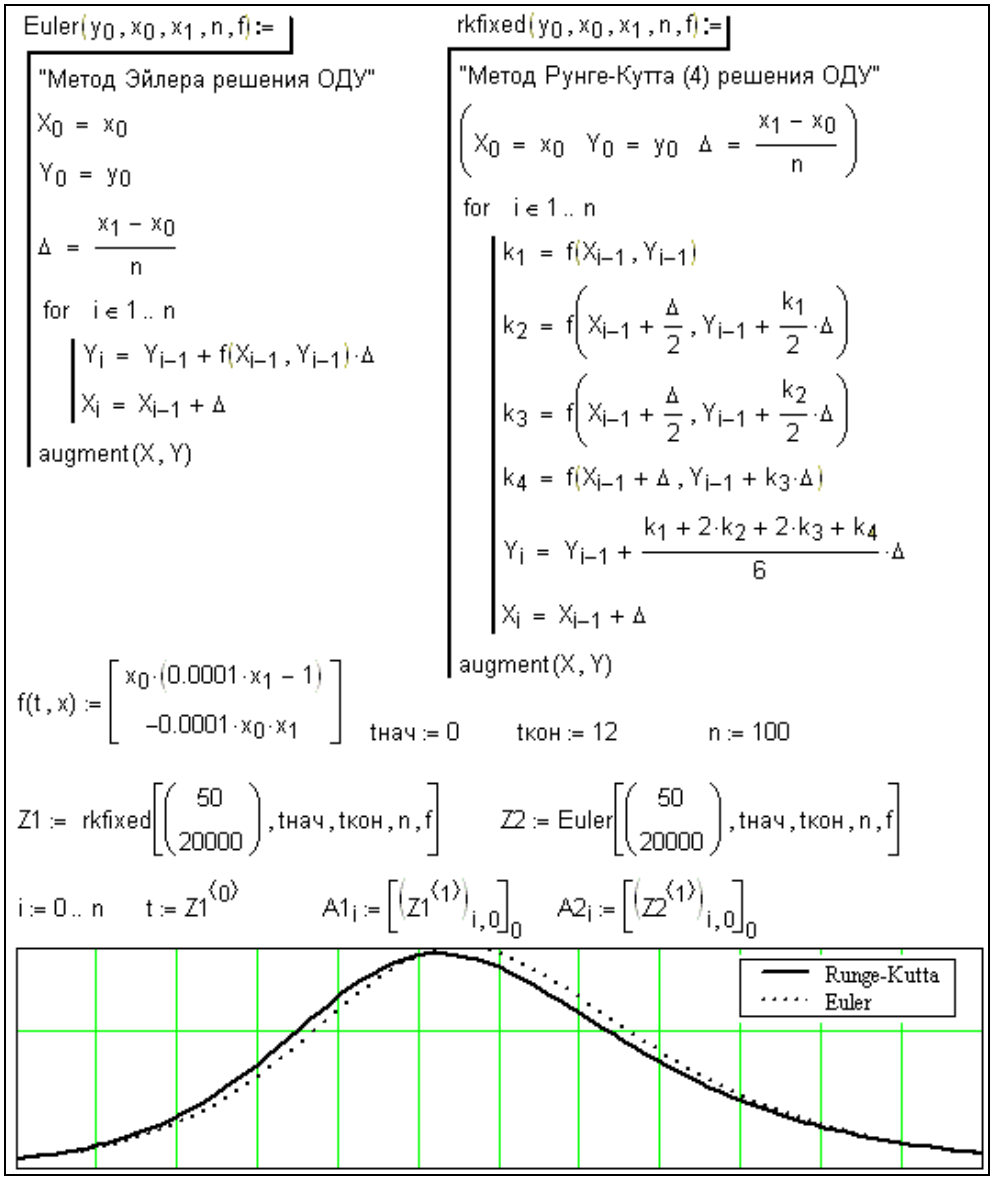


Рис. 5. Методы Эйлера и Рунге — Кутта в задаче об эпидемии

Автор не описывает работу остальных функций, введенных в 6-ю версию Mathcad для решения дифференциальных уравнений, по следующим причинам:

- ❒ достаточно полное описание дано в документации и справочной системе;
- ❒ многие отмеченные функции так и не прижились в расчетной практике;

- в последующих версиях Mathcad появились новые функции, во многом перекрывающие работу старых.

Отмечу только следующее. При решении *жестких систем* дифференциальных уравнений следует использовать одну из двух встроенных функций, разработанных специально для таких случаев: `Stiffb` и `Stiffr`. Они используют метод Булирша — Штера *(b)* или Розенброка *(r)*. Форма матрицы-решения, полученной с помощью этих функций, идентична матрице, полученной через `rkfixed`. Однако `Stiffb` и `Stiffr` требуют дополнительного аргумента `J`:

```
Stiffb(x, tнач, tкон, n, f, J)
Stiffr(x, tнач, tкон, n, f, J)
```

где:

- `x` — вектор `n` начальных значений;
- `tнач`, `tкон` — конечные точки интервала, в котором должно быть найдено решение дифференциальных уравнений. Начальные значения `x` определяются в точке `tнач`;
- `n` — количество точек за начальной точкой, в которых должно быть определено решение. Это определяет число рядов (`1+n`) матрицы, которую генерируют функции `Stiffb` и `Stiffr`;
- `f(t, x)` — функция-вектор правых частей системы;
- `J(t, x)` — матрица-функция размера `n·(n+1)`, в которой содержится матрица Якоби правых частей дифференциальных уравнений.

На рис. 6 показано сравнение работы функции `Stiffr` с функцией `rkfixed` при решении жесткой системы.

http://twt.mpei.ac.ru/MAS/Worksheets/rkfixad_Stiffr_Stiffb.mcd

$\alpha_0 := 0.002$ $\alpha_1 := 10$

$$DY(T,Y) := \begin{bmatrix} \alpha_0 \cdot Y_0 - Y_0 \cdot Y_1 \\ -\alpha_1 \cdot Y_1 + (Y_0)^2 \end{bmatrix} \quad J(T,Y) := \begin{pmatrix} 0 & \alpha_0 - Y_1 & -Y_0 \\ 0 & 2 \cdot Y_0 & -\alpha_1 \end{pmatrix}$$

$T_{beg} := 0$ $T_{end} := 50$ $N_{max} := 180$

$IC_0 := 0.6$

$IC_1 := 0.2$

$$IC := \begin{pmatrix} IC_0 \\ IC_1 \end{pmatrix}$$

MS := (•) rkfixed ( ) Stiffr

Solve Numerical

$$S := \begin{cases} \text{rkfixed}(IC, T_{beg}, T_{end}, N_{max}, DY) & \text{if } MS = \text{"rkfixed"} \\ \text{Stiffr}(IC, T_{beg}, T_{end}, N_{max}, DY, J) & \text{otherwise} \end{cases}$$

$T := S^{(0)}$ $Y_0 := S^{(1)}$ $Y_1 := S^{(2)}$

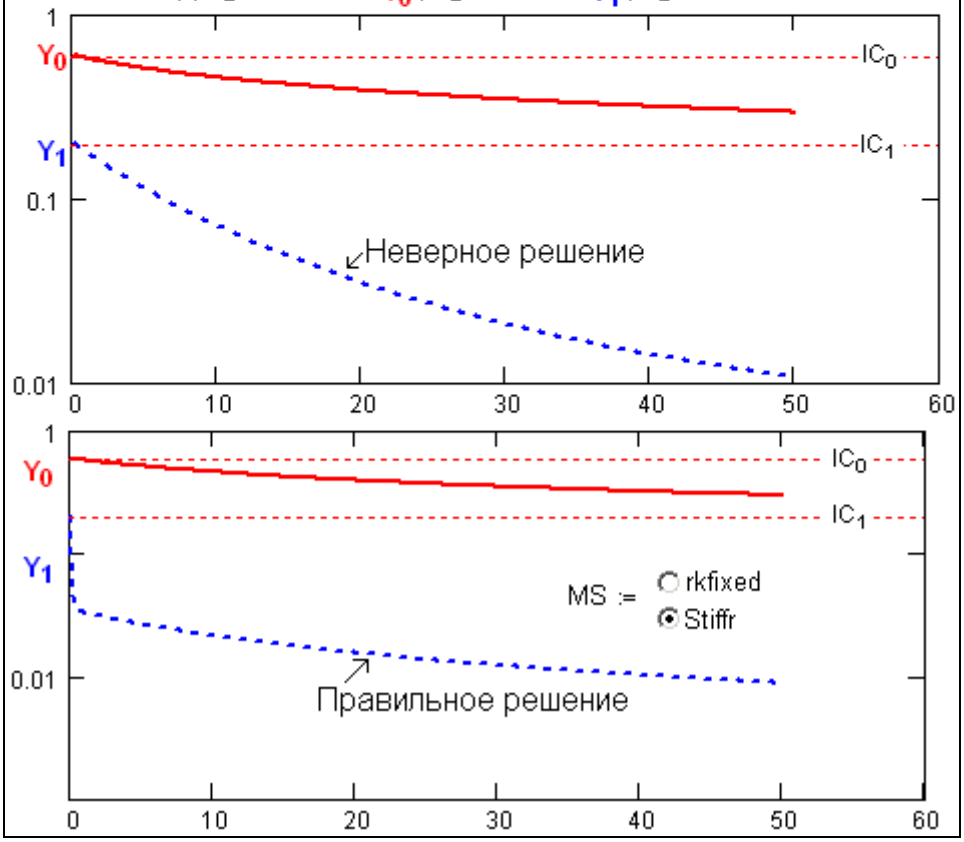


Рис. 6. Сравнение двух методов решения жесткой системы дифференциальных уравнений

Сравнение, правда, не совсем "честное". Стоит только изменить значение `n` с 180 на 200, например, как разница в расчетах сразу нивелируется: проверьте на MA/CS — **http://twt.mpei.ac.ru/MAS/Worksheets/rkfixad_Stiffr_Stiffb.mcd**.

В Mathcad 2000 появилась функция `odesolve`, предназначенная для решения (solve) обыкновенных дифференциальных уравнений (ode), и заодно решающая две побочные задачи — повышается наглядность решения и объединяются в одной функции разные алгоритмы.

Примечание

В Mathcad 2000 функция `odesolve` решала только одиночное дифференциальное уравнение, а в версии 2001 — уже и системы дифференциальных уравнений.

На рис. 7 показано, как функцию `odesolve` решает нашу задачу о развитии эпидемии.

Примечание

В Mathcad 2000 вместе с функцией `odesolve` появился оператор взятия производной в виде апострофа — `y′(x)`, который вводится в расчет нажатием комбинации клавиш <Ctrl>+<F7> и действует только между ключевым словом `Given` и функцией `odesolve`. Вне этих рамок данный апостроф превращается в простой комментарий.

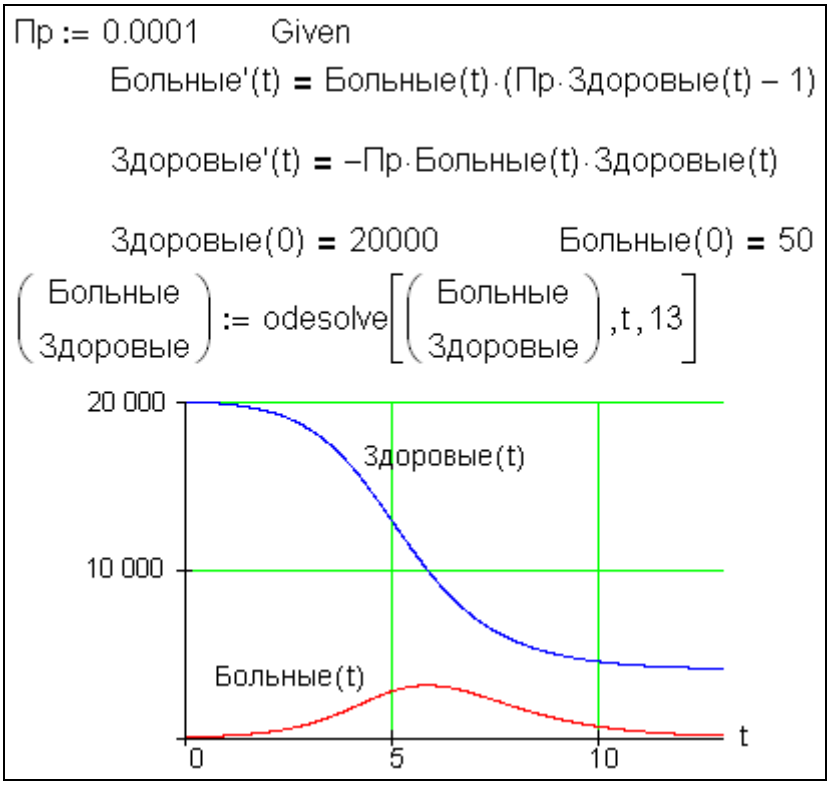


Рис. 7. Задача об эпидемии — решение системы дифференциальных уравнений с помощью функции `odesolve`

Функция `odesolve` уникальна в том, что она возвращает не *скалярное* или *векторное значение*, как другие встроенные функции Mathcad (`sin`, `cos` и т. д.), а новую *функцию*, у которой можно искать нули, максимумы и минимумы и осуществлять иные действия, допустимые к функциям. Все другие встроенные функции Mathcad[14], предназначенные для решения дифференциальных уравнений (`rkfixed` и др.), возвращают таблицу значений, по которым нужно формировать функцию, интерполяцией или сглаживанием. Но функция, генерируемая функцией `odesolve`, честно говоря, не совсем нормальная — к ней нельзя, например, применить операторы и команды символьных преобразований и некоторые другие операции. Это связано с тем, что функция, генерируемая функцией `odesolve`, создается той же интерполяцией по точкам.

Удобство функции `odesolve` в том, что она воссоздает естественную запись системы дифференциальных уравнений по схеме, которая в среде Mathcad уже давно используется для решения уравнений: `Given` (дано), сами уравнения и функция, возвращающая решение задачи. Роднит функции `Find` (`MinErr`, `Minimize`, `Maximize`) и

[14] Кроме функции `pdesolve`, которая появилась в Mathcad 11.

odesolve и то, что правая кнопка мыши вызывает локальное меню, в котором есть команды выбора алгоритма решения (рис. 8).

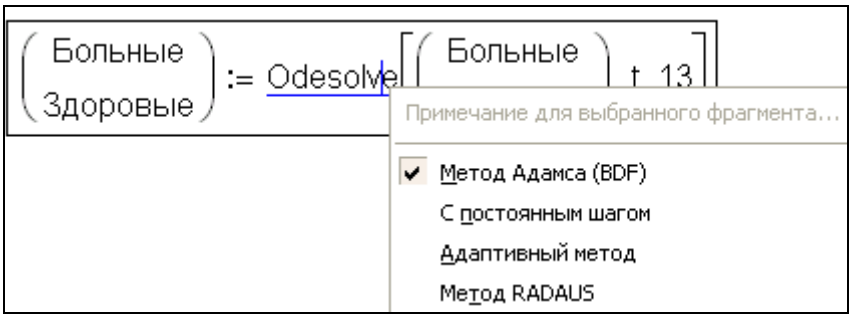


Рис. 8. Подстройка функции odesolve

На рис. 8 показано, что пользователь или сама система Mathcad может выбрать один из четырех алгоритмов решения задачи через функцию odesolve: **Метод Адамса (BDF[15]), С постоянным шагом**, **Адаптивный метод** и **Метод RADAUS**. В Mathcad 14 внесены изменения и добавления в инструменты решения обыкновенных дифференциальных уравнений. Во-первых, введена новая функция statespace (наряду с новыми функциями Бесселя (Dai, DBi, DAI.sc, Jacob и DBi.sc) и другими функциями), позволяющая решать ОДУ, записанные в матричной форме. Во-вторых, "старая" функция Radau получила три новых дополнительных аргумента, позволяющих влиять на точность и скорость ее работы. Кроме того, несколько изменен список алгоритмов, применяющихся при реализации функции odesolve (см. рис. 8). Теперь, например, при ориентации решения систем ОДУ по методу Адамса (BDF) пакет Mathcad будет "в динамике", по ходу самого численного интегрирования, сам решать, какой метод применить — для решения нежестких или жестких дифференциальных уравнений. Метод Адамса (BDF) оформлены также и в виде новых встроенных функций, что, в частности, позволяет встраивать их в Mathcad-программы *(см. далее)*.

На рис. 3.21 читатель может увидеть подобное меню настройки, но для функции Find. Такое же меню можно вызвать у функций MinErr, Minimize и Maximize.

Теперь несколько усложним нашу задачу об эпидемии. Если известно начальное количество больных (N=50 — см. рис. 1, 2 и 7), то это значит, что их сразу пересчитали. А если это так, то их знают поименно. Но в этом случае больные должны быть изолированы, и никакой эпидемии не будет вообще (Пр=0). В реальной задаче мы можем знать число жителей в городе (количество здоровых на день начала эпидемии) и число больных в какой-то последующий день, когда становится ясно, что разразилась эпидемия. Другими словами, нам что-то известно о параметрах на *краях* отрезка, охватывающего некий динамический процесс.

[15] Backward Differentiation Formula

На рис. 9 реализован метод последовательных приближений для решения по разностной схеме, отображенной на рис. 1, такой *краевой задачи* (а не задачи Коши): в начале эпидемии в городе 20 000 здоровых жителей, а в конце эпидемии — 100 больных. Спрашивается, сколько больных было в начале эпидемии?

ORIGIN := 1 $Здоровые_1 := 20000$ Пр := 0.0001 t := 1.. 13

Метод последовательных приближений

$Больные_1 := 40$

$$\begin{pmatrix} Больные_{t+1} \\ Здоровые_{t+1} \end{pmatrix} := \begin{pmatrix} Пр \cdot Здоровые_t \cdot Больные_t \\ Здоровые_t - Пр \cdot Здоровые_t \cdot Больные_t \end{pmatrix}$$

Перелет

$Больные_{13} = 163.9$

$Больные_1 := 60$

$$\begin{pmatrix} Больные_{t+1} \\ Здоровые_{t+1} \end{pmatrix} := \begin{pmatrix} Пр \cdot Здоровые_t \cdot Больные_t \\ Здоровые_t - Пр \cdot Здоровые_t \cdot Больные_t \end{pmatrix}$$

Недолет

$Больные_{13} = 72.61$

$Больные_1 := 50$

$$\begin{pmatrix} Больные_{t+1} \\ Здоровые_{t+1} \end{pmatrix} := \begin{pmatrix} Пр \cdot Здоровые_t \cdot Больные_t \\ Здоровые_t - Пр \cdot Здоровые_t \cdot Больные_t \end{pmatrix}$$

$Больные_{13} = 105.17$

$Больные_1 := 55$

$$\begin{pmatrix} Больные_{t+1} \\ Здоровые_{t+1} \end{pmatrix} := \begin{pmatrix} Пр \cdot Здоровые_t \cdot Больные_t \\ Здоровые_t - Пр \cdot Здоровые_t \cdot Больные_t \end{pmatrix}$$

$Больные_{13} = 86.73$

$Больные_1 := 52$

$$\begin{pmatrix} Больные_{t+1} \\ Здоровые_{t+1} \end{pmatrix} := \begin{pmatrix} Пр \cdot Здоровые_t \cdot Больные_t \\ Здоровые_t - Пр \cdot Здоровые_t \cdot Больные_t \end{pmatrix}$$

$Больные_{13} = 97.18$

$Больные_1 := 51$

$$\begin{pmatrix} Больные_{t+1} \\ Здоровые_{t+1} \end{pmatrix} := \begin{pmatrix} Пр \cdot Здоровые_t \cdot Больные_t \\ Здоровые_t - Пр \cdot Здоровые_t \cdot Больные_t \end{pmatrix}$$

$Больные_{13} = 101.06$

$Больные_1 := 51.326$

$$\begin{pmatrix} Больные_{t+1} \\ Здоровые_{t+1} \end{pmatrix} := \begin{pmatrix} Пр \cdot Здоровые_t \cdot Больные_t \\ Здоровые_t - Пр \cdot Здоровые_t \cdot Больные_t \end{pmatrix}$$

$Больные_{13} = 99.77$

Рис. 9. Решение краевой задачи об эпидемии разностной схемой

Ответ (51 больной) получен за семь приближений: задается начальное число больных, которое корректируется в зависимости от того, какое число больных оказывается в конце эпидемии. На рис. 9 можно, конечно, не дублировать Mathcad-операторы, а просто вручную подправлять первое (предыдущие) приближение.

С помощью функции `rkfixed`, решающей задачу Коши, краевую задачу можно также решить последовательными приближениями (рис. 10).

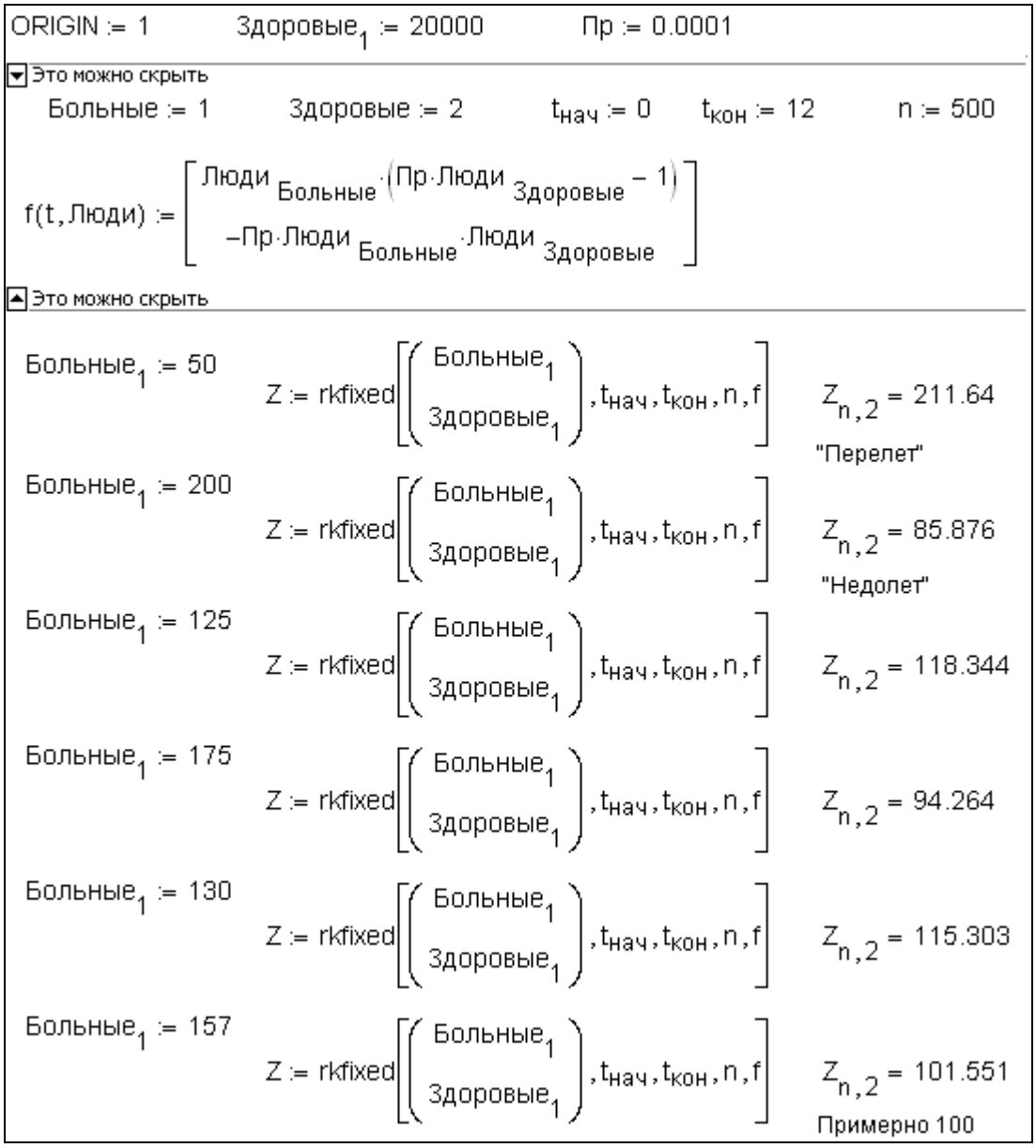


Рис. 10. Решение краевой задачи об эпидемии функцией `rkfixed`

Нижняя кривая на рис. 7 похожа на траекторию полета снаряда[16], поэтому метод последовательных приближений, приложенный к краевой задаче, называют также *ме-*

[16] Она может быть совсем не похожа на траекторию полета снаряда, но... читаем дальше.

тодом стрельбы: можно менять искомые начальные условия, последовательно приближаясь к решению, имея на другом конце отрезка "корректировщика огня", в лексиконе которого три слова: "перелет", "недолет" и "попал" ("почти попал", учитывая заданную точность расчета — см. комментарии на рис. 9). По правде говоря, метод стрельбы берет свое название не от вида кривой, а от особенностей решения краевой задачи применительно к дифференциальному уравнению второго порядка, когда приходится менять "угол наклона ствола пушки" — значение первой производной в начале отрезка интегрирования. В нашей задаче об эпидемии (система двух обыкновенных дифференциальных уравнений) для попадания в цель приходится менять не "угол наклона пушки", а ее "подъем" над землей — число больных в начале эпидемии. Тут, как правило, используют метод половинного деления, когда отрезок "перелет-недолет" делят пополам.

Метод стрельбы заложен и во встроенную в Mathcad функцию `sbval`, работа которой показана на рис. 11. Она требует начального приближения и соблюдения некоторых условностей, связанных с нашими знаниями состояний решаемой системы на концах отрезка интегрирования. Функция `sbval` не совсем обычная. Мы привыкли к тому, что, например, операторы `A:=sin(X)` и `A:=sin(30)` идентичны, если переменная `X` имеет значение `30`, несмотря на то, что в первом случае у функции `sin` в качестве аргумента выступает переменная, а во втором — константа. Функция же `sbval` в этом отношении аномальна. Она возвращает значение в зависимости не только от конкретных значений аргументов, но и от того, введены они в виде переменных (x_0) или в виде констант (`20 000`). Из-за этого даже специалистам по дифференциальным уравнениям часто приходится ломать голову, чтобы сообразить, как можно условия краевой задачи "запихнуть" в функцию `sbval`. Здесь фирма Mathsoft несколько отошла от первоначальной идеологии пакета Mathcad, подразумевавшей, что запись условия задачи на экране дисплея должна выглядеть естественно.

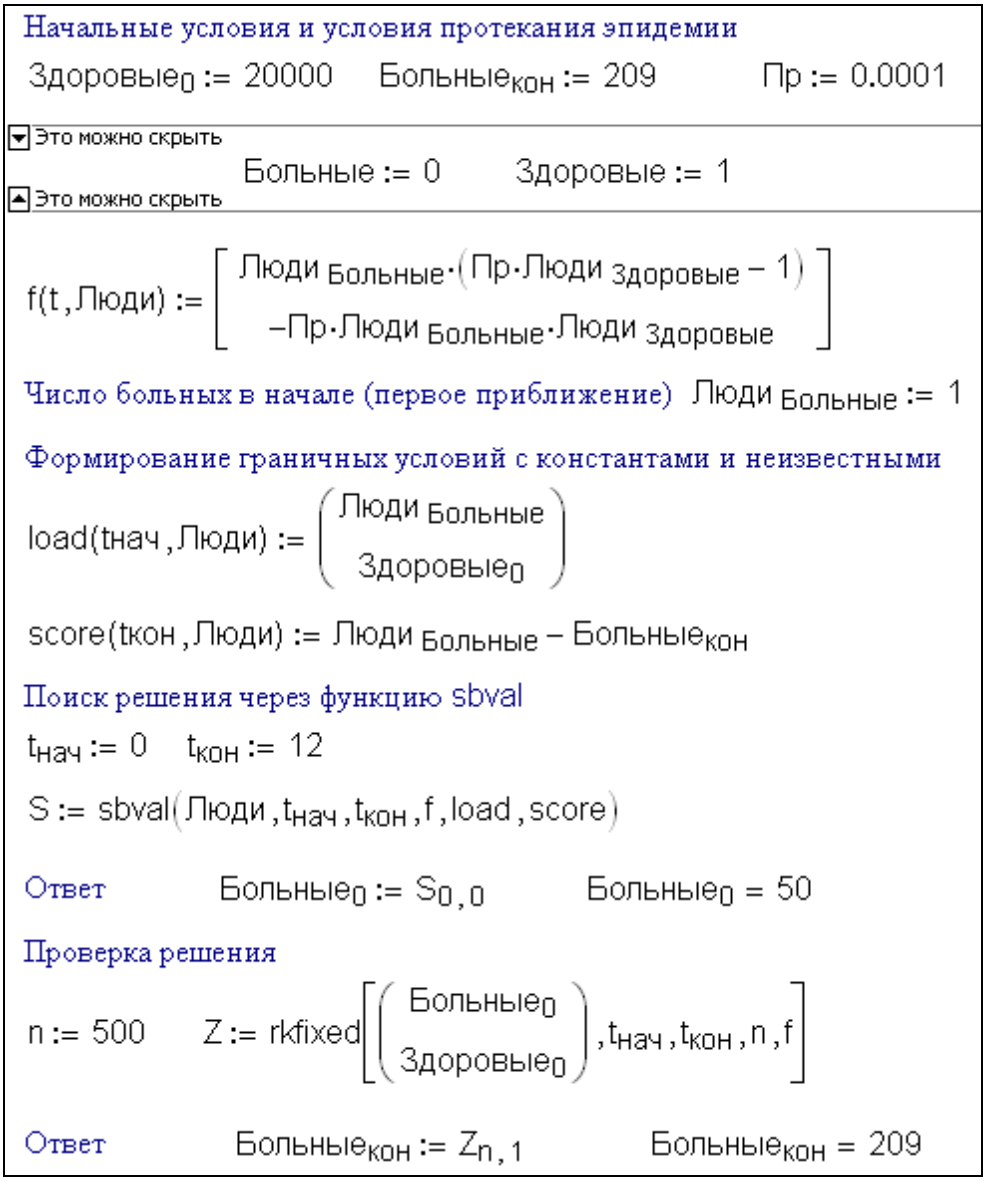


Рис. 11. Решение краевой задачи об эпидемии функцией `sbval`

Типичный пример — численное решение в среде Mathcad системы алгебраических уравнений: сначала задается начальное приближение к корню (ведь корней может быть много), потом за ключевым словом `Given` (дано) в естественном виде пишется сама система уравнений, после которой помещается функция `Find`, возвращающая значения своих аргументов, превращающих ранее записанные уравнения (до слова

Given) в тождества. Функция Find также аномальна: она возвращает значения, зависящие не только от значений аргументов, но и от того, что вблизи нее находится. Это, конечно, грубейшее нарушение правил построения функциональных зависимостей, и это уже отмечено в *разд. 1.2.1*. Но разработчикам Mathcad пришлось пойти на это ради того, чтобы алгебраические уравнения в Mathcad-документе хранились в естественной форме. При создании математических пакетов типа Mathcad приходится решать не только чисто математические проблемы (разработка алгоритмов решения и реализация их средствами вычислительной математики), но и проблемы единства *формы* представления задачи (интерфейс пользователя) и методов ее решения (*содержание* задачи).

Функция sbval не решает краевую задачу, а только находит недостающие значения на краю отрезка. После этого краевая задача переходит в задачу с начальными условиями (задача Коши), которая для проверки и решается так, как показано внизу рис. 11.

Для решения краевой задачи в среде Mathcad есть еще одна функция — bvalfit. Она используется в тех случаях, когда нет всей необходимой информации для функции sbval, но известно решение задачи в промежуточной точке. Функция bvalfit решает задачу с двумя граничными точками, начиная с конечных точек и следуя траекториям решения и его производным в промежуточных точках:

```
bvalfit(x1, x2, tнач, tкон, tf, f, load1, load2, score)
```

где:

- x1 — вектор предполагаемых начальных значений, не определенных в точке x1;
- x2 — то же для точки x2;
- tнач, tкон — конечные точки интервала, в котором должно быть вычислено решение дифференциального уравнения;
- tf — точка между tнач и tкон, в которой траектории решений, начатые с tнач, и траектории, начатые с tкон, должны совпадать;
- f(t, x) — векторная функция, состоящая из n элементов и содержащая первые производные неизвестной функции;
- load1(tнач, x1) — векторная функция, n элементов которой соответствуют значениям n неизвестных функций в точке x1. Некоторые из этих значений будут постоянными, заданными начальными условиями. Другие будут неизвестны вначале, но будут найдены. Если значение неизвестно, то следует использовать соответствующее предполагаемое значение из вектора x1;
- load2(tкон, x2) — аналог load1, но для значений n неизвестных функций в точке x2;
- score(tf, x) — векторная функция, состоящая из n элементов. Эта функция применяется для того, чтобы задать, как решения должны совпадать в точке tf. Обычно нужно определить score(tf, x):=x, чтобы все решения неизвестных функций совпадали в точке tf.

Функция `bvalfit` по своей сути решает две краевые задачи на двух смежных отрезках интегрирования. Она особенно полезна, когда производная имеет разрыв где-то внутри интервала интегрирования. При развитии эпидемии (наша задача) такое может произойти, если, например, комиссия из центра снимет с работы медицинское начальство города и тем самым резко изменит значение коэффициента `Пр`.

На рис. 12 зафиксирован еще один подход к решению краевой задачи "Эпидемия", без использования сложных и довольно туманных встроенных функций `sbval` и `bvalfit`.

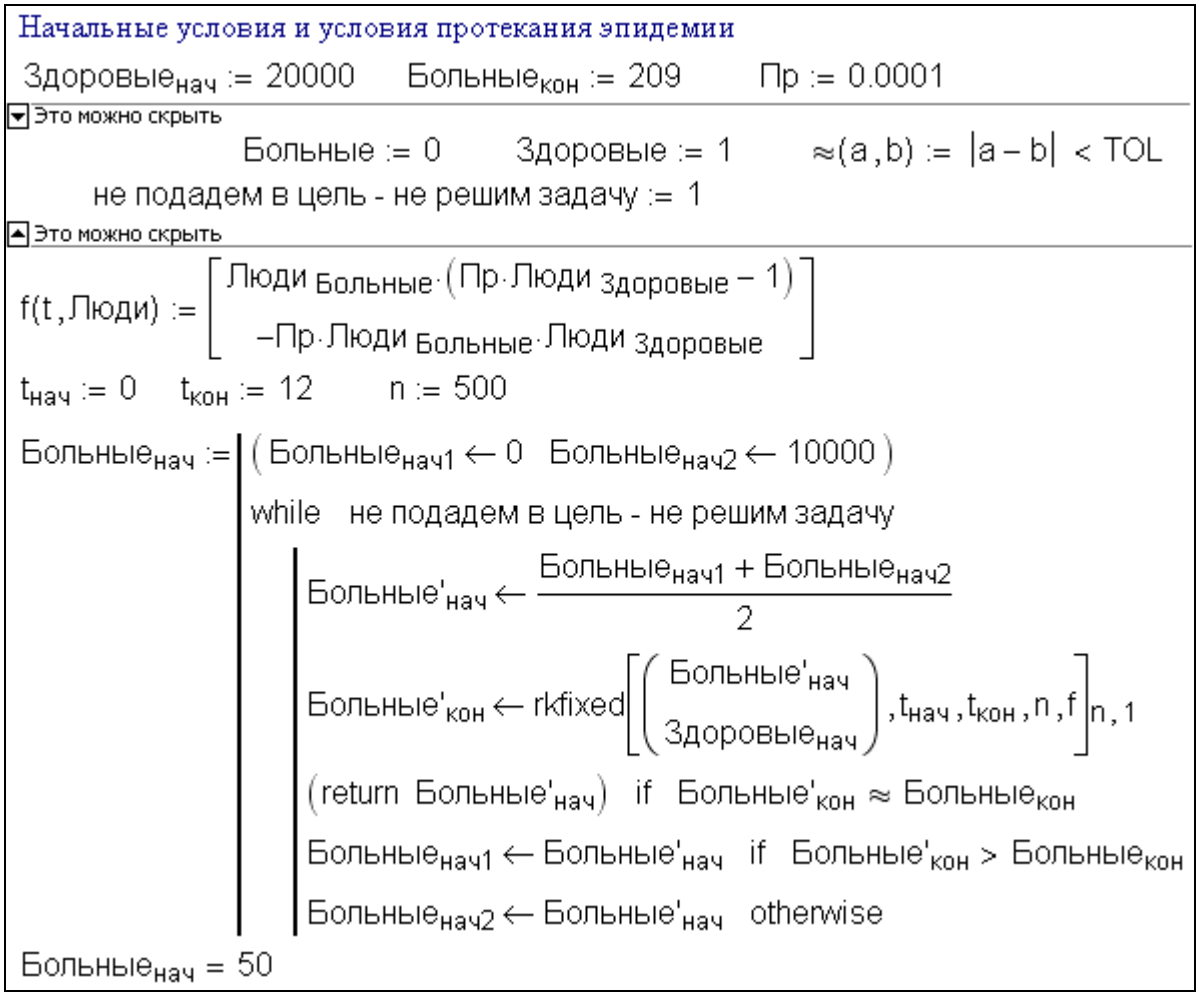


Рис. 12. Решение краевой задачи об эпидемии средствами программирования

На рис. 12 с помощью средств программирования Mathcad автоматизированы последовательные приближения при реализации метода стрельбы и метода половинного деления, "ручной" вариант которых был показан на рис. 10.

Средства программирования Mathcad позволяют также построить фазовый портрет нашей задачи об эпидемии (рис. 13: кривые при различных значениях начального числа больных).

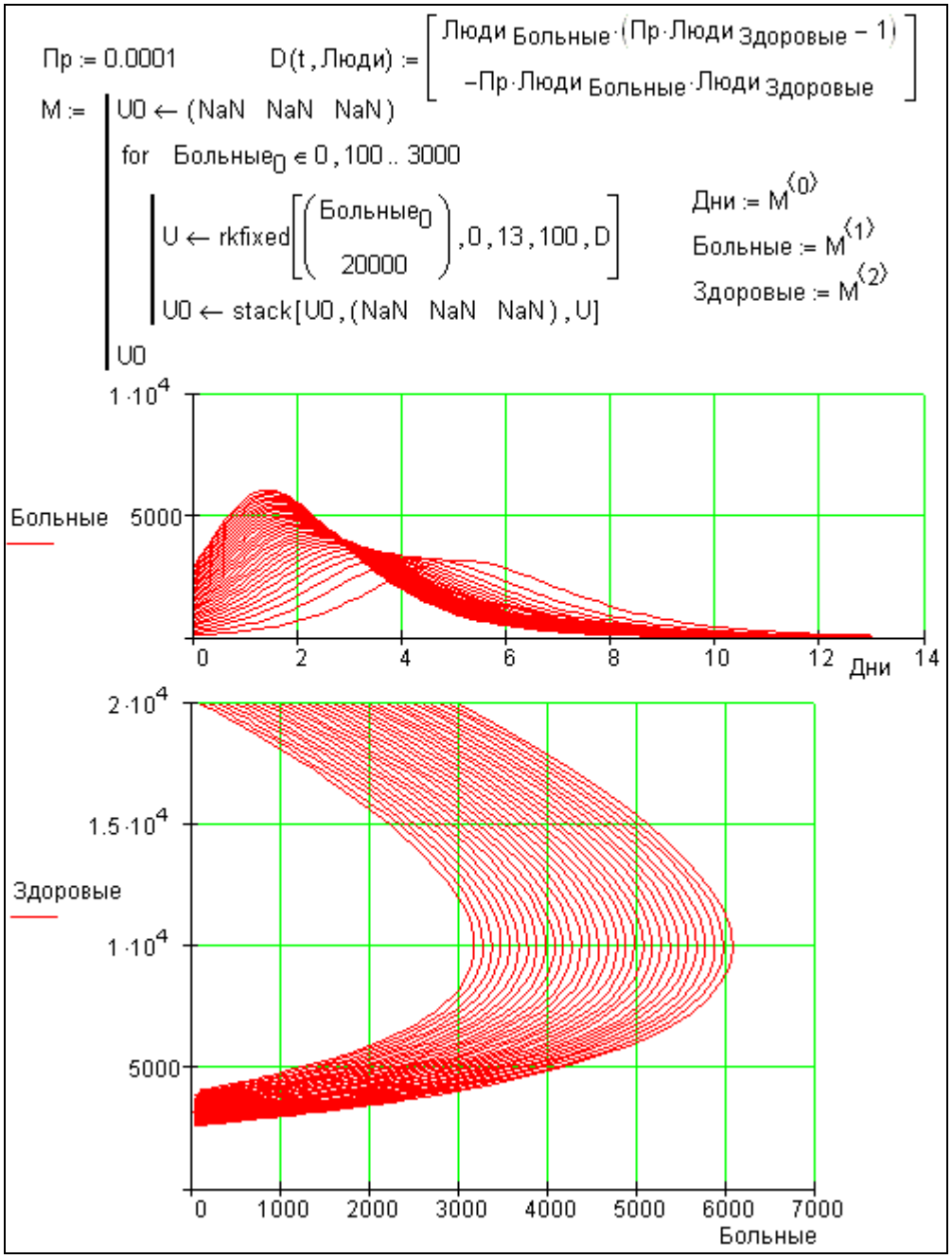


Рис. 13. Фазовый портрет задачи об эпидемии

Казалось бы, что подстраиваемая (см. рис. 8) функция `odesolve`, введенная в Mathcad 2000 и расширенная в Mathcad 2001 и 14, должна "похоронить" все остальные функции, оставив их только для совместимости — для того, чтобы Mathcad-документ, написанный в 6-й версии, например, работал и в 14-й. Но это не совсем так. Основной недостаток функции `odesolve` в том, что ее нельзя вставить в программы. Этому мешает ключевое слово `Given`. И если возникнет необходимость программно реализовать метод стрельбы (половинного деления) для решения краевой задачи, то тут нужно будет вернуться к функции `rkfixed` (см. рис. 12), например, несмотря на все другие вышеописанные преимущества функции `odesolve`.

Но и "первородная" функция `rkfixed` и другие подобные функции имеют свои недостатки. Основной из них — невозможность создания функции пользователя для многократного вызова этой функции при формировании различного вида зависимостей — новых пользовательских функций. Такую задачу можно решить только за счет некоего обмана пакета Mathcad. Так на рис. 14 (первая его половина) сделана попытка ввода в функцию `f`, хранящую правые части системы двух дифференциальных уравнений развития эпидемии[17], дополнительного аргумента `Пр` — `f(t, x, Пр)`. Но ничего из этого не вышло. Было выдано сообщение об ошибке "This function needs more arguments" ("Эта функция требует больше аргументов"), хотя здесь аргументов должно быть наоборот меньше.

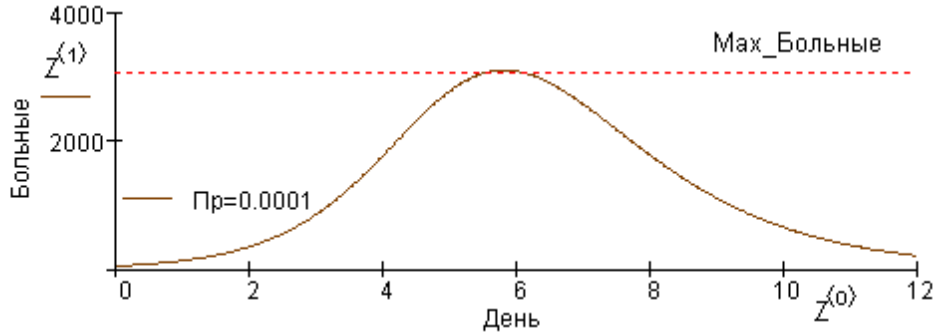
[17] Здесь мы вернулись к записи X_0 и X_1 вместо более длинных, но более понятных $Люди_{Больные}$ и $Люди_{Здоровые}$ (см. рис. 6.2).

$Здоровые_0 := 20000$ $Больные_0 := 50$ $Пр := 0.0001$

$t_{beg} := 0$ $t_{end} := 12$ $n := 500$

$$f(t,x) := \begin{bmatrix} x_0 \cdot (Пр \cdot x_1 - 1) \\ -Пр \cdot x_0 \cdot x_1 \end{bmatrix} \qquad Z := rkfixed\left[\begin{pmatrix} Больные_0 \\ Здоровые_0 \end{pmatrix}, t_{beg}, t_{end}, n, f\right]$$

$Max\_Больные := \max\left(Z^{\langle 1 \rangle}\right)$ $Max\_Больные = 3119$



Попытка получения функции пользователя с решеним

$$f(t,x,Пр) := \begin{bmatrix} x_0 \cdot (Пр \cdot x_1 - 1) \\ -Пр \cdot x_0 \cdot x_1 \end{bmatrix}$$

$$Z(Пр) := rkfixed\left[\begin{pmatrix} Больные_0 \\ Здоровые_0 \end{pmatrix}, t_{beg}, t_{end}, n, f\right]$$

$Z(0.0001) =$ ▪▪

Эта переменная не определена

Эта функция имеет форму: *f*(Unitless, [Unitless], Unitless) → [Unitless], а должна иметь форму: *f*(Unitless, Unitless) → Unitless.

Одно из решений этой проблемы

$$F(t,x) := \begin{bmatrix} x_0 \cdot (x_2 \cdot x_1 - 1) \\ -x_2 \cdot x_0 \cdot x_1 \\ 0 \end{bmatrix} \qquad Z(Пр) := rkfixed\left[\begin{pmatrix} Больные_0 \\ Здоровые_0 \\ Пр \end{pmatrix}, t_{beg}, t_{end}, n, F\right]$$

$Пр := 0.000000, 0.00001 .. 0.001$

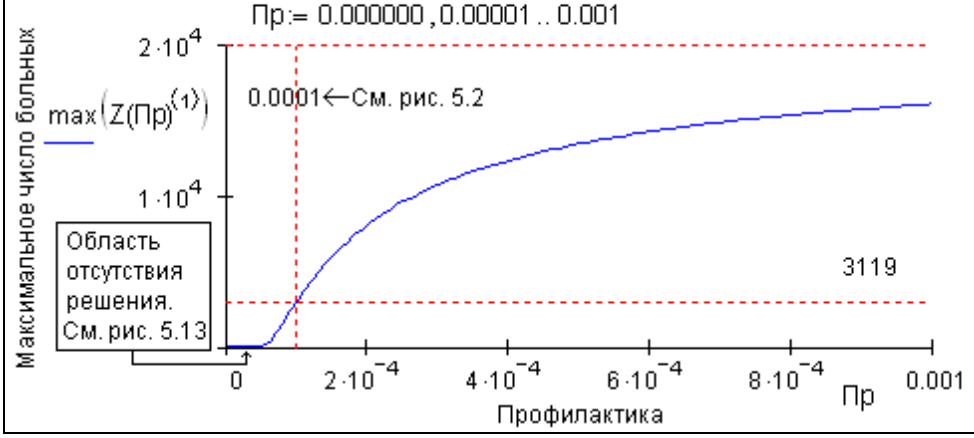


Рис. 14. Создание функции пользователя на базе встроенной функции `rkfixed`

Положение спасло добавление в функцию-вектор `F(t, x)` дополнительного не аргумента, а элемента (компонента вектора) с нулевым (пустым) значением. После этого матрицу решения задачи `Z` стало возможным заменить на функцию `Z(Пр)` и построить "одним махом", например, график изменения максимального количества больных (вершина "горы", показанной на рис. 2 и 7) от значения коэффициента профилактики (см. рис. 14 внизу).

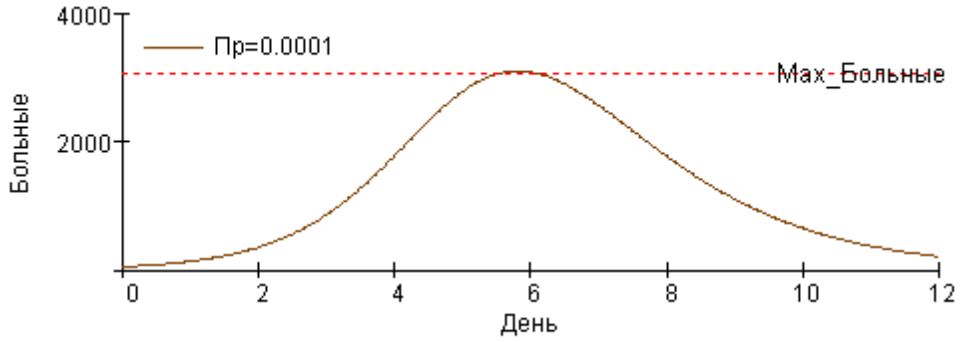
В среде Mathcad 12/13/14 появилась возможность создания локальной функции, что позволяет нам более естественно решить задачу о создании функции пользователя на базе встроенной `rkfixed` (рис. 15)

$Здоровые_0 := 20000$ $Больные_0 := 50$ $Пр := 0.0001$

$t_{beg} := 0$ $t_{end} := 12$ $n := 500$

$$f(t,x) := \begin{bmatrix} x_0 \cdot (Пр \cdot x_1 - 1) \\ -Пр \cdot x_0 \cdot x_1 \end{bmatrix} \qquad Z := rkfixed\left[\begin{pmatrix} Больные_0 \\ Здоровые_0 \end{pmatrix}, t_{beg}, t_{end}, n, f\right]$$

$Max\_Больные := \max\left(Z^{\langle 1 \rangle}\right)$ $Max\_Больные = 3119$



$$Z(Пр) := \left| \begin{array}{l} f(t,x) \leftarrow \begin{bmatrix} x_0 \cdot (Пр \cdot x_1 - 1) \\ -Пр \cdot x_0 \cdot x_1 \end{bmatrix} \\ rkfixed\left[\begin{pmatrix} Больные_0 \\ Здоровые_0 \end{pmatrix}, t_{beg}, t_{end}, n, f\right] \end{array} \right.$$

$Пр := 0.000000, 0.00001 .. 0.001$

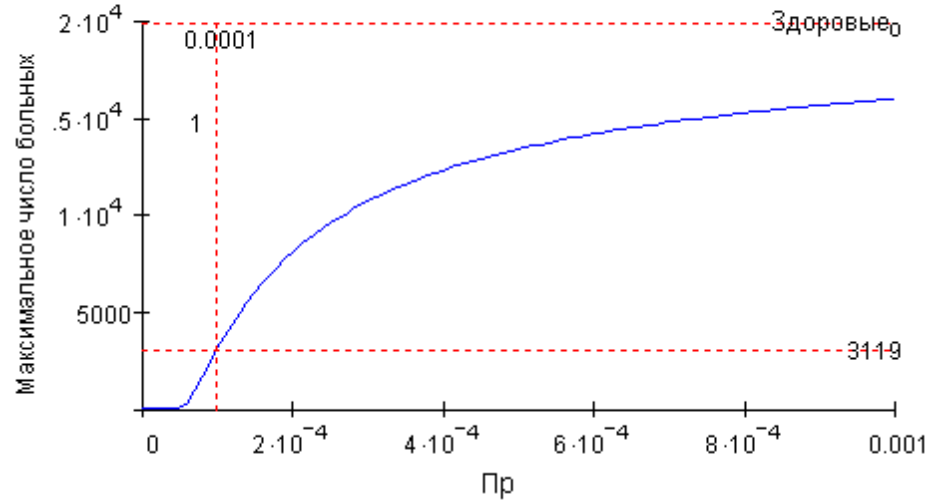


Рис. 15. Создание функции пользователя на базе встроенной функции `rkfixed` с использованием локальной функции

Встроенные инструменты Mathcad имеют еще один общий недостаток, связанный с тем, что при возникновении ошибки счета не дается полной информации о причинах сбоя. Из-за этого часто приходится отказываться от встроенных средств и возвращаться к пользовательским, "доморощенным" функциям. Так "в пику" встроенным в Mathcad функциям `Find` и `Minimize`, мы описали функции поиска корней систем уравнений и минимумов функции средствами программирования (см. рис. 4.35, например), которые при всей их примитивности имеют одно преимущество — они выдают *траекторию* пути к решению.

Встроенные в Mathcad средства решения дифференциальных уравнений также обладают этим недостатком. Очень часто решения систем дифференциальных уравнений прерываются сообщением об ошибке, связанной с переполнением предела счета (10^{307}). Такое случилось с нашей задачей об эпидемии когда значения коэффициента профилактики `Пр` заменили с 0.0001 на 0.00885, например. При этом не ясно, какое именно уравнение системы "зашкалило", как нужно его подправить или как следует изменить пределы интегрирования, чтобы получить хотя бы упрощенное или укороченное решение для его последующего анализа. Такая ситуация отображена на рис. 16, где (первая половина) функция `rkfixed` дает сбой.

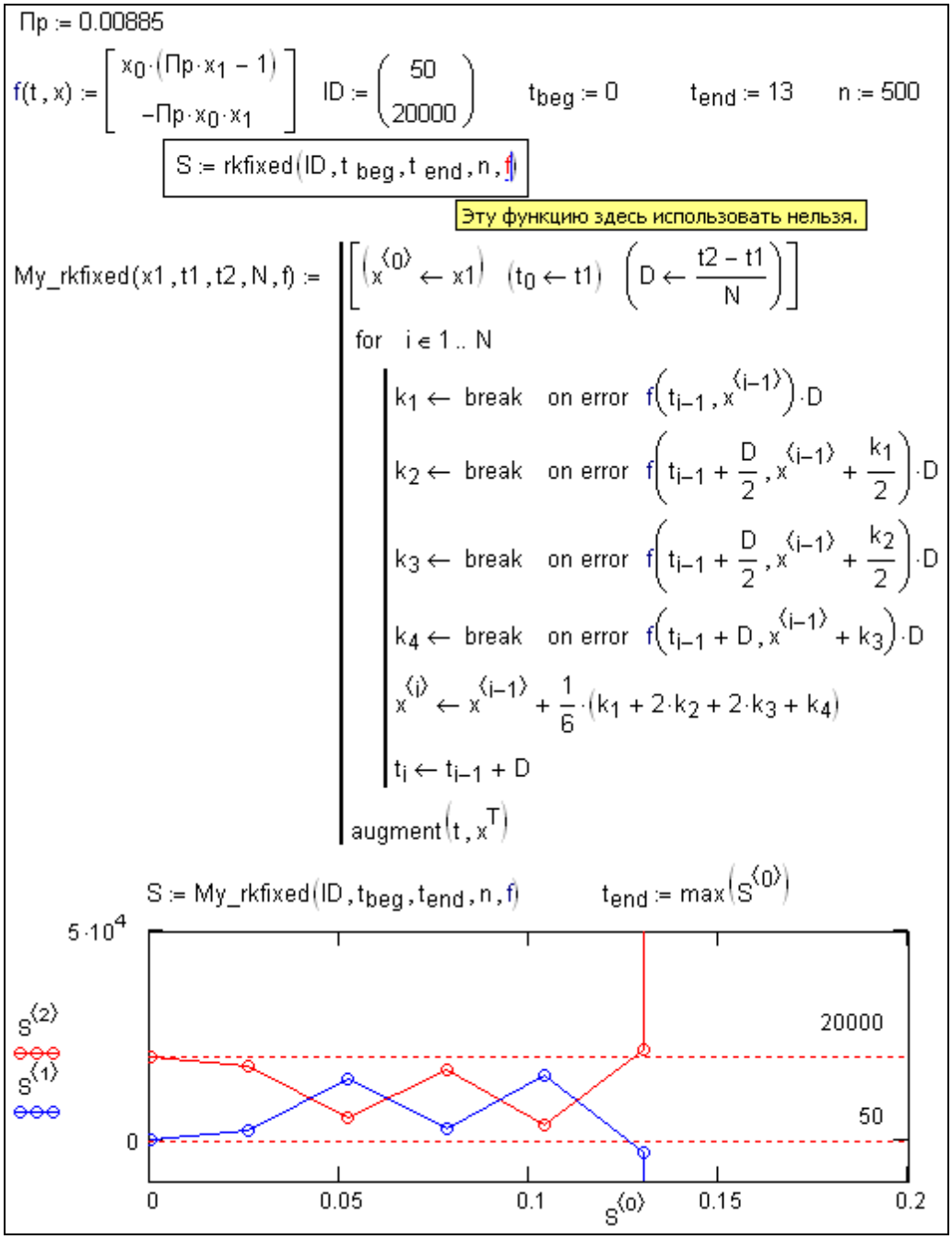


Рис. 16. Замена встроенной функции `rkfixed` на пользовательскую

Во второй половине рис. 16 встроенная функция `rkfixed` заменена на программу, реализующую метод Рунге — Кутта и возвращающую при ошибке *частичное* решение задачи за счет операторов обработки ошибок `on error`, вставленных в программу. Это позволяет вести интегрирование не до конечной точки (у нас это 3), а до последней допустимой. Встроенная же функция же `rkfixed` излишне категорична — она возвращает "все или ничего".

В среде Mathcad 13/14 встроенные средства отладки позволяют провести трассировку и встроенных средств решения ОДУ (рис. 17).

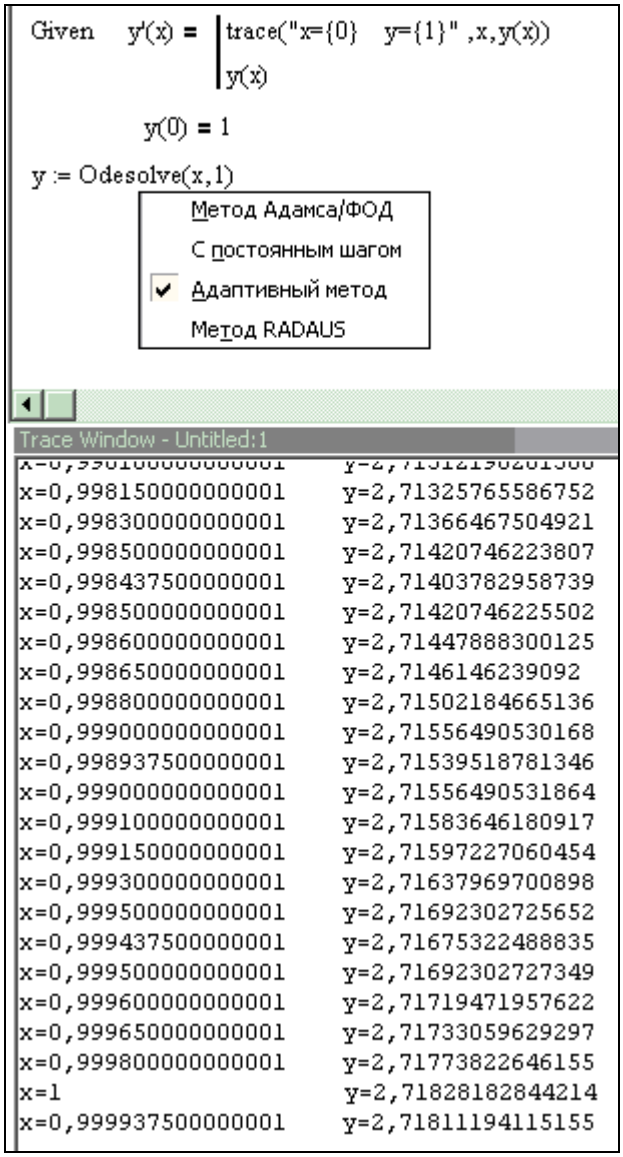


Рис. 17. Трассировка встроенной функции `odesolve`