Глава 10. Полет на Луну

Так назывался советский мультфильм 1953 года, два кадра которого показаны на рис. 1. Его можно посмотреть в интернете. Фильм наивный, но довольно забавный, если принять во внимание, что через четыре года был запущен первый искусственный спутник Земли. А в 1961 году человек полетел в космос. А в 1969 году люди высадились на Луне.

В те далекие времена почти все мальчишки, включая и автора этой книги, мечтали стать космонавтами и так рисовали старт космического аппарата: ракета разгоняется по длинной параболической эстакаде и по диагонали взмывает ввысь «навстречу звездам» – см. рис. 1.

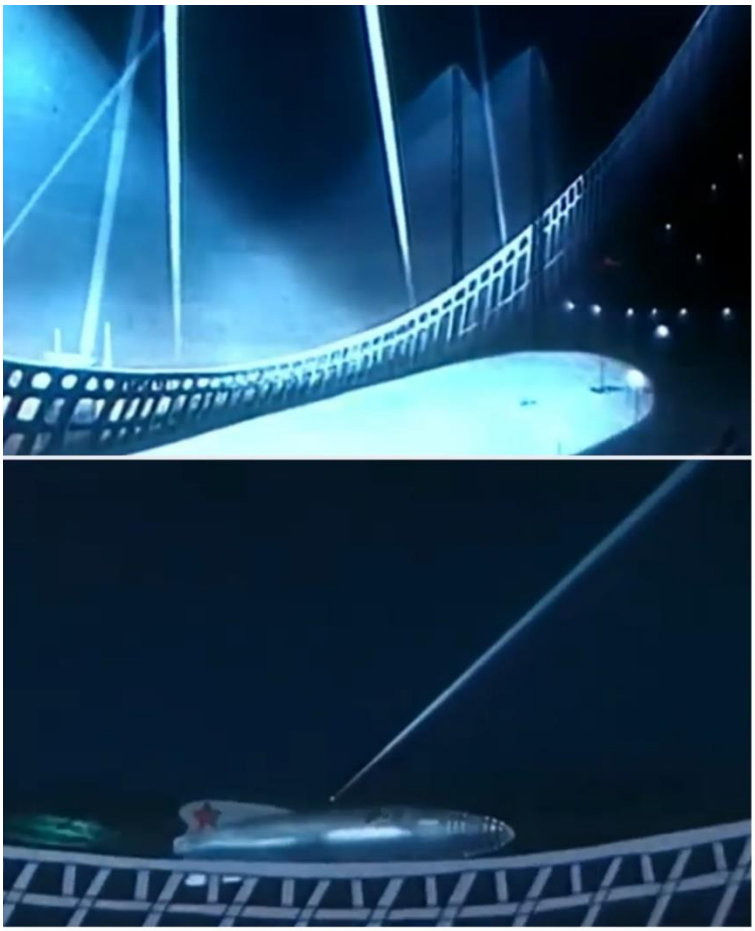


Рис. 1. Два кадра из мультфильма «Полет на Луну»

Но когда автор, будучи уже не ребенком, а юношей впервые увидел кадры кинохроники пуска реальной ракеты, то он очень удивился и разочаровался. Никакой романтики! Ракета, привязанная к какому-то столбу, отвязывается от него и как бы нехотя отрывается от поверхности Земли и медленно в клубах дыма поднимаясь вверх (см. рис. 8 ниже). Дело в том, что до конца 60-х годов прошлого века в Советском Союзе по соображениям секретности кинохронику пуска ракет не показывали ни в кино, ни по телевизору. Поэтому-то ребята и рисовали фантастические огромные эстакады из знаменитого в те времена мультфильма.

А давайте рассчитаем пуск ракеты – изменение во времени ее высоты и скорости. Заодно покажем, зачем ракеты по совету Циолковского делают многоступенчатыми.

На рисунке 2 показан такой расчет в среде SMath для одноступенчатой ракеты. Исходные данные (операторы с ярлыками-комментариями) условные. Читатель может ими «поиграть» и посмотреть, что будет получаться.

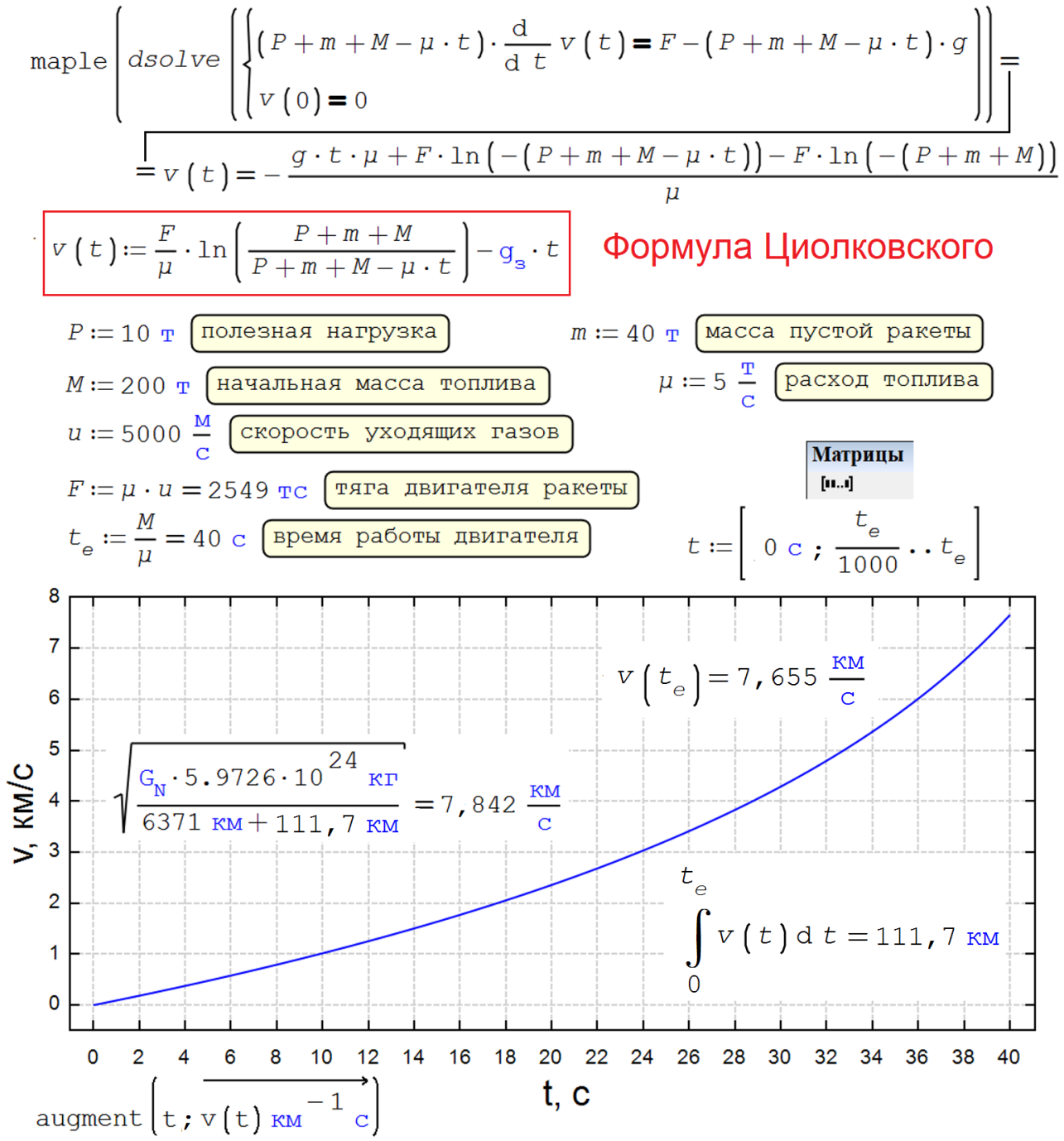


Рис. 2. Расчет полета одноступенчатой ракеты

На рисунке 2 до ввода исходных данных (числа с единицами измерения) аналитически решается дифференциальное уравнение (частный случай уравнения Мещерского), отображающее второй закон

Ньютона. Масса материальной точки (физического объекта, имеющего массу, но не имеющего размера) со временем уменьшается за счет сгорания топлива и окислителя. Эта масса умножается на ускорение – на первую производную скорости v по времени t (левая часть уравнения). На ракету действуют две силы в противоположных направлениях – тяга ракетного двигателя (F) и сила притяжения Земли (правая часть уравнения). Первая сила постоянная, а вторая уменьшается за счет сгорания топлива. Константа g ($g_з$) – это ускорение свободного падения. Кстати, в советском мультфильме можно видеть, что двигатель фантастической ракеты импульсный. У нас же ракета с двигателем непрерывного действия.

В среде SMath нет средств аналитического решения дифференциальных уравнений. Они есть в пакете Maple, который как плагин подключается к SMath через команду меню Сервис/Дополнения/Галерея онлайн. Выданное функцией dsolve решение вручную дорабатывается и формирует функцию пользователя $v(t)$, которая отображается на графике. Мы получили знаменитую формулу Циолковского с натуральным логарифмом отношения массы ракеты в двух моментах времени.

Внутри графика на рисунке 2 помещены три расчетные оператора, из которых видно, что наша ракета поднимется на высоту почти 112 км, но не достигнет первой космической скорости и не выведет на орбиту искусственный спутник Земли. Формула первой космической скорости взята из интернета, но ее при желании тоже можно вывести. Для вычисления пройденного пути используется определенный интеграл.

А как все-таки вывести спутник на орбиту Земли!? Можно увеличивать запас топлива и окислителя, а можно поступить иначе – запустить в ближний космос двухступенчатую ракету, использующую такое же или даже меньшее количество топлива и окислителя.

Расчет, показанный на рис. 3, отличается от предыдущего тем, что некоторые исходные данные представлены не в виде скаляров, а в виде векторов с двумя элементами, хранящими параметры двух ступеней ракеты. Кроме того, введены две функции-ступеньки, возвращающие массу ракеты $m(t)$ и ее тягу $F(t)$ в зависимости от времени. В эти функции введен оператор if, который затруднит аналитическое решение задачи по шаблону, показанному на рис. 2. Можно, конечно, разбить задачу на две части и найти отдельно формулы Циолковского для стартовых двух ступеней и для второй ступени ракеты. Но если в дальнейшем учитывать сопротивление воздуха и изменение по высоте ускорения свободного парения, то аналитика станет бессильной. Нашу задачу нужно будет решать числено, а не аналитически. А каким методом?

В настоящее время наблюдается некий вычислительный ренессанс [1]. Пользователи современных быстродействующих компьютеров возвращаются к простым и понятным алгоритмам. Раньше они применялись редко в том числе и из-за медленного счёта старых ЭВМ. В настоящее время это ограничение существенно ослаблено в первую очередь для задач учебного плана, где недоговоренность нежелательна.

Вернемся в славную эпоху запуска первых космических аппаратов. В те времена происходило не только «покорение космоса», но и наблюдался революционный переход от ручных расчетов к расчетам на электронных вычислительных машинах – цифровых и аналоговых. Если говорить о цифровых электронных вычислительных машинах (ЭЦВМ – иностранное слово компьютер тогда избегали употреблять), то они работали по программам, использующим различные алгоритмы, в том числе и по тем, какие были предложены несколько веков назад. Началась гонка не только в космосе, но и на Земле – создавались все более точные и быстрые программы решения тех же дифференциальных уравнений, отображающих полет космических аппаратов. Самый первый и самый простой метод численного решения дифференциального уравнения в 1768 году предложил великий Эйлер (1707–1783). В это время он работал в России. Но этот метод как-то ушел в тень. Его оттеснили всякие

Рунге, Кутты, Адамсы, Левенберги, Марквардты… К сожалению, фамилий советских ученых в этом списке нет. Отчасти повторилась старая история с уравнением Мещерского (см. ниже), усугубившаяся условиями секретности[1].

У нас компьютер быстрый, да и задача довольно простая. Давайте решим ее методом Эйлера!

Его суть предельно проста и в этом его единственное преимущество. Нам известно значение искомой функции (скорости полета ракеты) в предыдущей точке – мы можем оценить значение этой функции в последующей точке, если нам известно значение производной искомой функции (ускорения ракеты) в предыдущей точке. Этот метод можно было бы назвать методом касательных (вернее, методом касательной – см. рис. 4), если бы это название не было бы вторым названием метода Ньютона [1], предназначенного для численного решения алгебраических, а не дифференциальных уравнений.

В настоящее время скорость даже такого компьютера как смартфон[2], несравнимо выше скорости первых компьютеров. Это и определяет отмеченный ренессанс в вычислениях. Усложненные быстрые алгоритмы и программы имеют один существенный недостаток – их трудно понять непосвященному. Это вносит некий дискомфорт в процесс решения задачи. А тут должна быть радость, удовольствие и удовлетвореннось.

На рисунке 3 сразу за функциями `m(t)` и `F(t)` введена функция `a(t)` – производная искомой функции `v` (a: acceleraton – ускорение). Далее записана небольшая программа с циклом `for`, генерирующая матрицу `tv` (время и скорость), из которой затем изымаются векторы `t` и `v` для построения графика изменения скорости ракеты по высоте.

Конечная скорость двухступенчатой ракеты (10,91 км/с) с запасом превысила первую космическую. Ура! Спутник вышел на орбиту

[1] На закате Советской власти выдавали талоны за сданную макулатуру для приобретение дефицитных книг, многие из которых, кстати, сами были макулатурой. Говорят, что в эти застойные времена в утиль отнесли много пакетов перфокарт и мотков перфолент с ценнейшими программами для ЭВМ.

[2] Программа SMath может работать и на смартфоне.

$P := 10\ т$ $m := [\ 40\ \ 5\ ]\ т$ $M := [\ 160\ \ 40\ ]\ т$ $\mu := [\ 5\ \ 2\ ]\ \frac{т}{с}$ $u := 5000\ \frac{м}{с}$

$\Delta t := \frac{\overrightarrow{M}}{\mu} = [\ 32\ с\ \ 20\ с\ ]$ $t_e := \sum \Delta t = 52\ с$

$$m(t) := P + \text{if } t < \Delta t_1 \left(\sum (M + m)\right) - \mu_1 \cdot t \text{ else } M_2 + m_2 - \mu_2 \cdot \left(t - \Delta t_1\right)$$

$$F(t) := u \cdot \text{if } t < \Delta t_1 \mu_1 \text{ else } \mu_2$$

$$a(t, v) := \frac{F - (P + m + M - \mu \cdot t)\, g_з}{P + m + M - \mu \cdot t}$$

Вставка матрицы: Строки: 1, Столбцы: 3

Матрицы

Y[1

$Euler(y_b; x_b; x_e; n; a) :=$ (y.b)

$X_1 := x_b$ $Y_1 := y_b$ $\Delta := \frac{x_e - x_b}{n}$

for $i \in [2..(n+1)]$

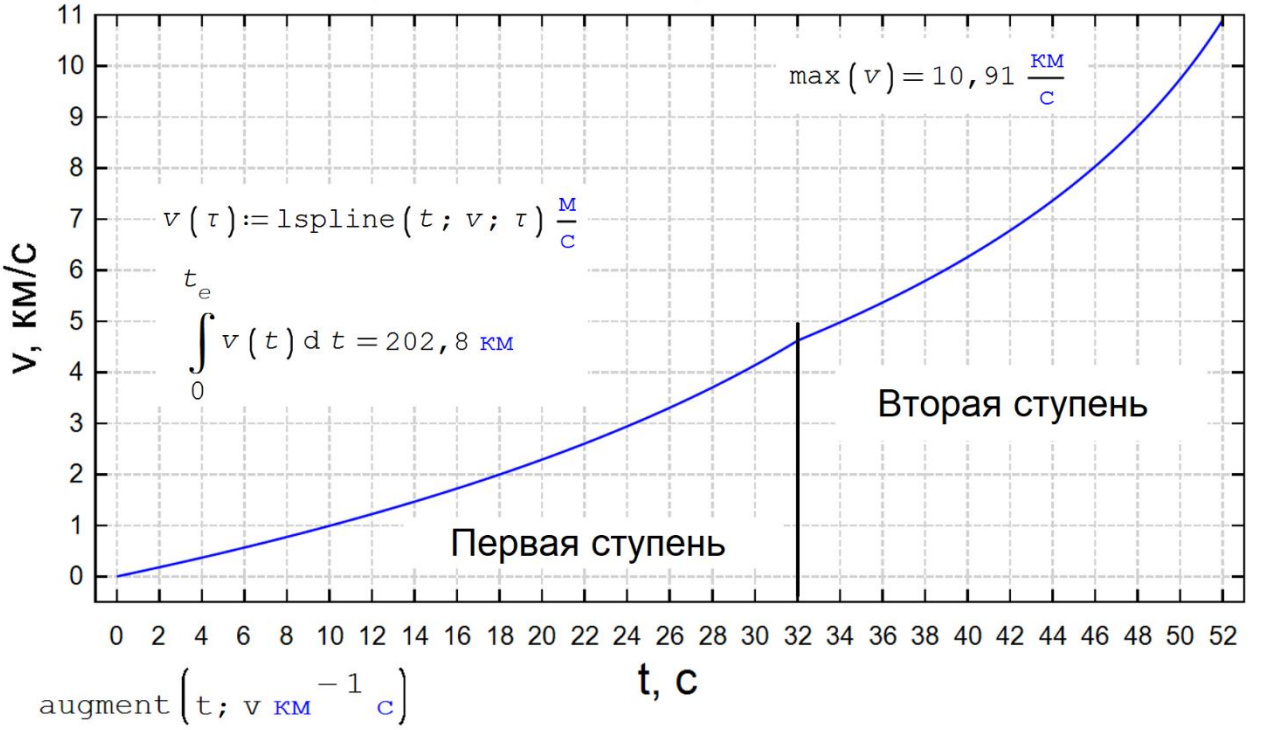
$Y_i := Y_{i-1} + a(X_{i-1}, Y_{i-1}) \cdot \Delta$

$X_i := X_{i-1} + \Delta$

augment$(X; Y)$

Программирование: if line for

$n := 1000$ $tv := Euler\left(0\ \frac{м}{с}; 0\ с; t_e; n; a\right)$ $t := \text{col}(tv; 1)$ $v := \text{col}(tv; 2)$



augment$(t; v\ км^{-1}\ с)$

Рис. 3. Расчет полета двухступенчатой ракеты

Встроенная в SMath функция `lspline` проводит сплайн-интерполяцию (см. главу 6) дискретных значений, хранящихся в векторах t и v. По полученной функции $v(t)$ через интегрирование рассчитывается высота полета. Такой же расчет через интеграл был сделан и для одноступенчатой ракеты (рис. 2), но там функция $v(t)$ была сразу получена через аналитическое решение задачи и интерполяции не потребовалось.

На рисунке 4 показаны результаты работы функции *Euler* с одноступенчатой ракетой, когда известно аналитическое – абсолютно точное решение. Под синей гладкой кривой (аналитическое решение) прорисованы так называемые ломанные Эйлера – приближения (аппроксимации) численного решения к аналитическому при n = 2, 10 и 300. В решении на рисунке 3 значение n равно 1000, что гарантирует приемлемую точность решения. Использование n=10000 (см. верхний левый угол на рис. 4) – это уже «ловля блох», как говорят прикладные математики.

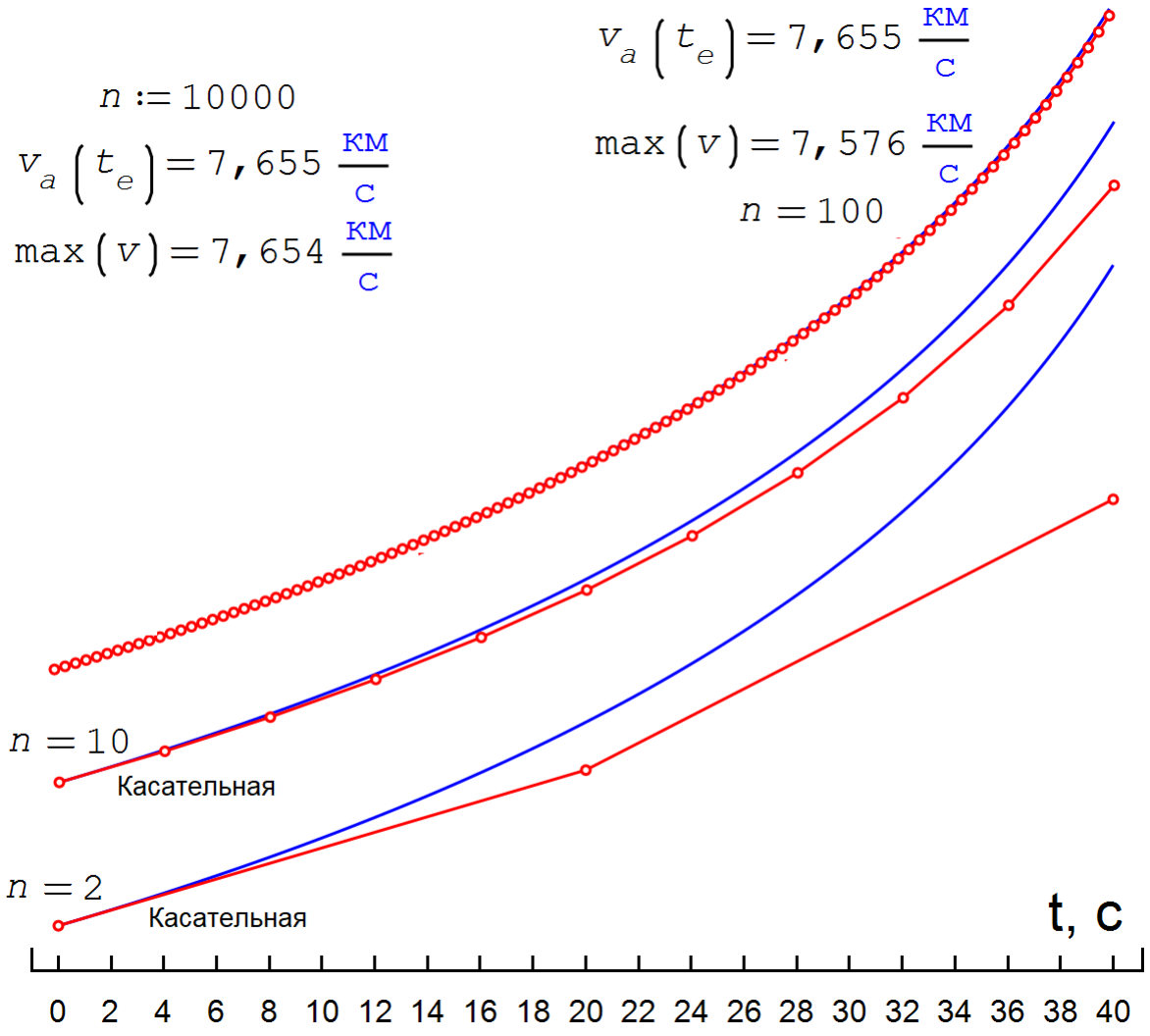


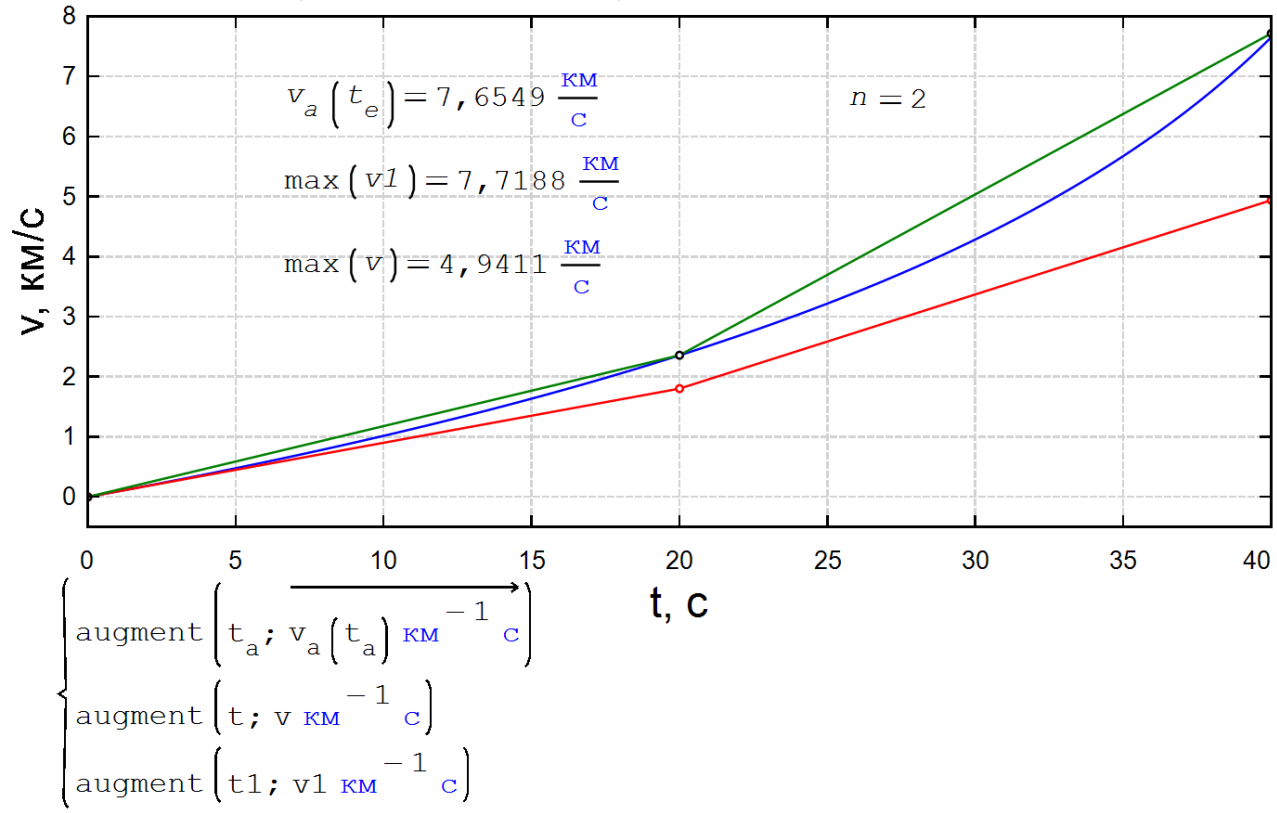
Рис. 4. Ломанные Эйлера

Можно заметить, что функция пользователя с именем a на рис. 3 имеет два аргумента – переменные t и v, хотя достаточно одного аргумента t. Эту особенность мы упомянем ниже.

Около 1900 года немецкими математиками Карлом Рунге (1856-1927) и Мартином Куттой (1867-1944) был предложен метод, существенно ускоряющий численное решение обыкновенных дифференциальных уравнений, увеличивающий точность. Он показан на рисунке 5.

$$RungeKutta\left(y_b;\ x_b;\ x_e;\ n;\ a\right):=\left|\begin{array}{l}\left[X_1:=x_b \quad Y_1:=y_b \quad \Delta:=\frac{x_e-x_b}{n}\right]\\ \text{for } i\in\left[2..(n+1)\right]\\ \quad\left|\begin{array}{l}k_1:=a\left(X_{i-1};\ Y_{i-1}\right)\\ k_2:=a\left(X_{i-1}+\frac{\Delta}{2};\ Y_{i-1}+\frac{\Delta}{2}\cdot k_1\right)\\ k_3:=a\left(X_{i-1}+\frac{\Delta}{2};\ Y_{i-1}+\frac{\Delta}{2}\cdot k_2\right)\\ k_4:=a\left(X_{i-1}+\Delta;\ Y_{i-1}+\Delta\cdot k_3\right)\\ Y_i:=Y_{i-1}+\frac{\Delta}{6}\cdot\left(k_1+2\cdot k_2+2\cdot k_3+k_4\right)\\ X_i:=X_{i-1}+\Delta\end{array}\right.\\ augment\left(X;\ Y\right)\end{array}\right.$$

$$tv1:=RungeKutta\left(0\ \frac{м}{с};\ 0\ с;\ t_e;\ n;\ a\right) \quad t1:=col\left(tv1;\ 1\right) \quad v1:=col\left(tv1;\ 2\right)$$



$$\left\{\begin{array}{l}augment\left(\overrightarrow{t_a;\ v_a\left(t_a\right)}\ км^{-1}\ с\right)\\ augment\left(t;\ v\ км^{-1}\ с\right)\\ augment\left(t1;\ v1\ км^{-1}\ с\right)\end{array}\right.$$

Рис. 5. Сравнение методов Эйлера и Рунге – Кутты (n = 2)

В расчет вводятся четыре коэффициента k_1, k_2, k_3, и k_4 для некой «пристрелки» – для стрельбы не на расстояние Δ, а не его половину. Результат такого приема показан на рис. 5 и 6.

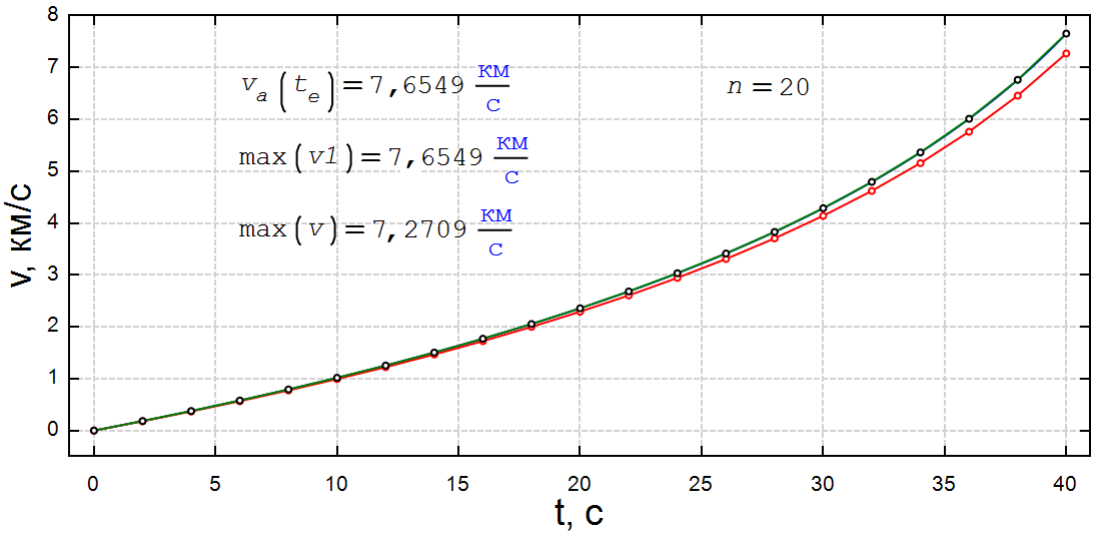


Рис. 6. Сравнение методов Эйлера и Рунге – Кутты (n = 20)

В среде SMath можно не писать функцию пользователя, реализующей метод Рунге-Кутта, а воспользоваться встроенной функцией с именем `rkfixed`. Буквы fixed означают, что «ходьба» вдоль аргумента искомой функции ведется с фиксированным (постоянным) шагом. Его можно менять, если работать с функцией `rkadapt`, подстраивающей (адаптирующей шаг) поиска решения. Это дополнительно ускоряет расчет, делает его более точным. Об этих нюансах можно прочесть на форуме пользователей SMath по адресу https://en.smath.com/forum/yaf_poststt726_Mathcad-Toolbox.aspx.

Ремарка. В решениях, показанных на рисунках 3 и 5, используются инструменты, собранные в панели «Программирование».

Когда говорят о программировании, то выделяют три его атрибута:

1. Объединение отдельных операторов в программные блоки, которые выполняются в отдельные операторы – см. вертикальные прямые линии в программах на рис. 3 и 5.
2. Локальные переменные, видимые только в программе. Это, например, переменные k_1, k_2, k_3, и k_4 в программе на рис. 5.
3. Структурные управляющие конструкции, изменяющие естественный порядок выполнения операторов слева направо и сверху вниз. В программах на рис. 3 и 5 использовалась конструкция for (цикл с параметром), заставляющая два (рис. 3) или шесть (рис. 5) операторов тела цикла выполняться несколько раз.

На рисунке 7 показан расчет полета двухступенчатой ракеты с использованием встроенной, а не пользовательской функции, реализующей метод Рунге – Кутты. Преимущество встроенной функции в том, что запись дифференциального уравнения ведется в естественной форме без выделения в правой части уравнения производной искомой функции. Более того, можно решать уравнения не только первого порядка, и не только одиночные уравнения, а и их системы. Недостаток же встроенной функции в том, что она не может работать с физическими величинами. Приходится эту функцию «обманывать» – делать единицы измерения безразмерными: см. третий аргумент функции `rkfixed`, записанный под дифференциальным уравнением второго порядка и начальными условиями. В этом

третьем аргументе число 1000 – это переменная n – число разбиений интервала, где табулируется искомая функция.

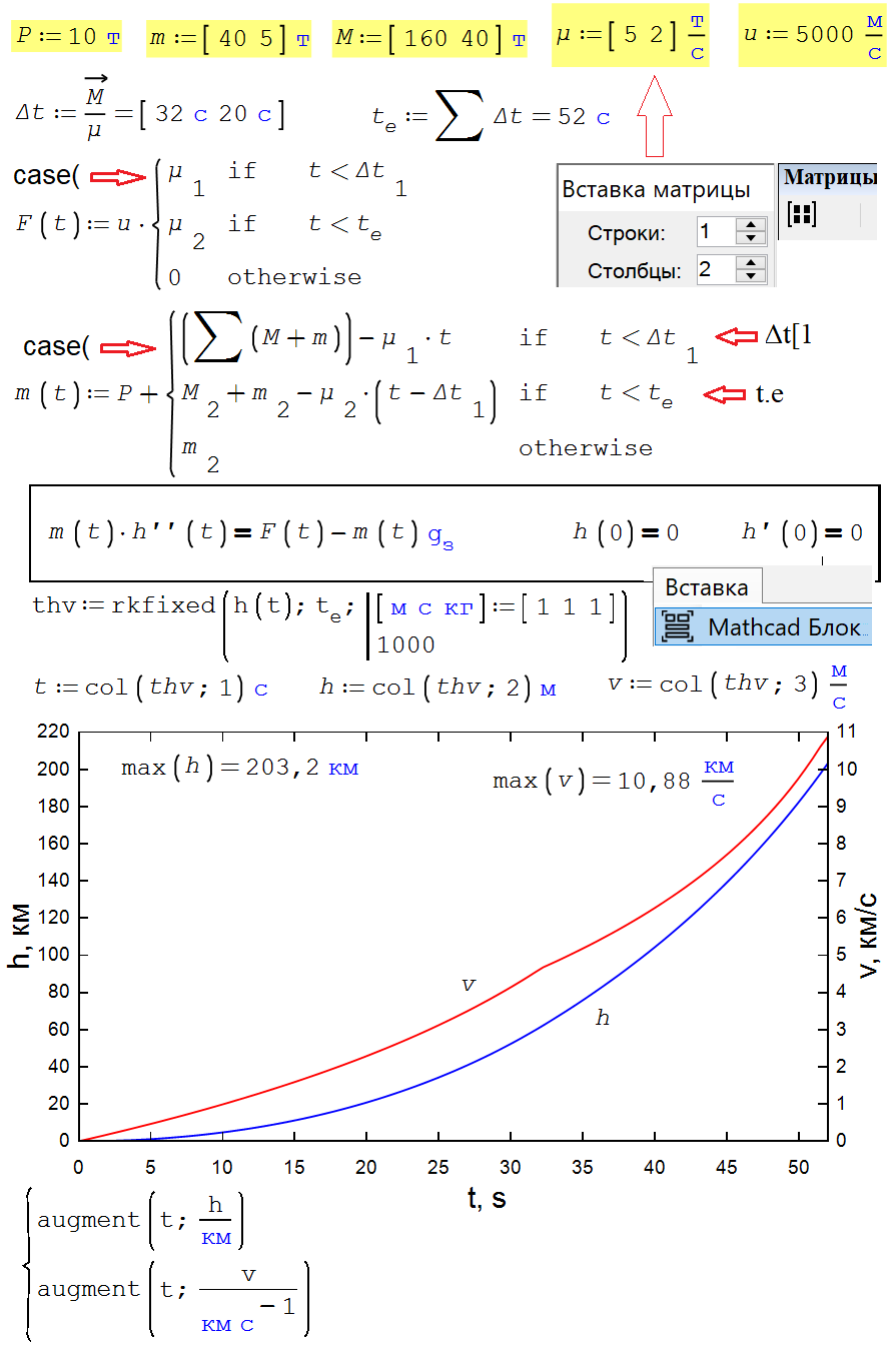


Рис. 7. Работа с Mathcad-блоком и с функцией rkfixed

Конечно, траектория полета ракеты, выводящей на орбиту ИСЗ, далеко не вертикальная прямая – см. рис. 8. Но наши расчеты можно усложнить – ввести, например, два дифференциальных уравнения, с балансом сил по вертикали и горизонтали.

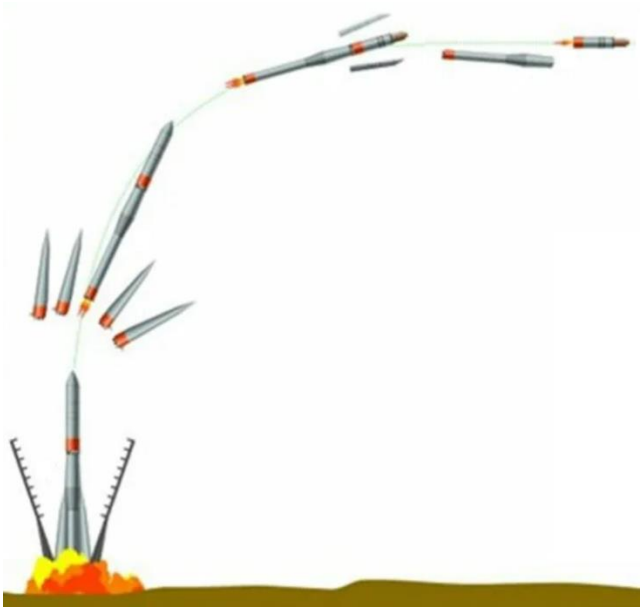


Рис. 8. Примерная траекторий вывода ИСЗ на орбиту

На сайтах пользователей SMath по адресам https://en.smath.com/forum/yaf_poststst24863_Error-by-ODEs-solution.aspx и https://en.smath.com/forum/yaf_postsm84488_Euler-method-without-for.aspx обсуждалась задача о полете ракеты. Там, в частности, рассматривалось аналитическое решение с выводом формулы Циолковского и другие интересные моменты – например, разная трактовка второго закона Ньютона применительно к телу с переменной массой (масса, умноженная на первую производную скорости по времени или первая производная произведения массы тела на его скорость – на импульс тела). Там же можно скачать решения, показанные выше.

На отмеченных сайтах рассматривался и более совершенный метод численного решения дифференциального уравнения – метод Рунге – Кутта (https://ru.wikipedia.org/wiki/Метод_Рунге_—_Кутты).

На рисунках 2 и 3, как было уже отмечено, приведены частные случаи уравнения Мещерского (https://ru.wikipedia.org/wiki/Уравнение_Мещерского). Дифференциальное уравнение движения тела переменной массы, которое И.В. Мещерский вывел и опубликовал ещё в 1897 году, спустя 31 год было (по незнанию трудов Мещерского) заново выведено итальянским математиком Туллио Леви-Чивита. К сожалению, как это уже не раз бывало, о русском первооткрывателе на Западе забыли, и это уравнение стали называть уравнением Леви-Чивита.

Литература:

1. Очков В.Ф., Чудова Ю.В. Вычислительный ренесанс: метод Ньютона // Энергия: экономика, техника, экология. № 9. 2023. С. 7-16 (http://www.twt.mpei.ac.ru/ochkov/EEE-9-2023-Newton.pdf)