**Лабораторная работа №1**

**МАТЕМАТИЧЕСКИЕ МОДЕЛИ ЛИНЕЙНЫХ СИСТЕМ**

**АВТОМАТИЧЕСКОГО УПРАВЛЕНИЯ**

1. ЦЕЛЬ РАБОТЫ

Ознакомиться с пакетом моделирования MatLAB. Освоить основные приемы моделирования систем автоматического управления.

1. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Поведение динамической линейной системы автоматического управления может быть описано скалярным дифференциальным уравнением n-го порядка



(1)

где y – выходная переменная; u – входной сигнал; m – порядок производной входного сигнала; ai и bj – постоянные коэффициенты.

Уравнение (1.1) можно записать в виде системы уравнений первого порядка



(2)

где xi – координаты вектора состояния, αij и βi – постоянные коэффициенты.

Система уравнений (1.2) может быть представлена в компактной векторно-матричной форме



(3)

где *A – n*х*n* – мерная матрица постоянных коэффициентов системы; *В – n*х*1* – мерная матрица постоянных коэффициентов входа; *c – 1*х*n* – мерная матрица постоянных коэффициентов выхода; *X* – *n*-мерный вектор состояния.

1. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

4.1. Ознакомиться с пакетом прикладных программ MATLAB (см. Приложение А).

4.2 Ознакомиться с методом интегрирования обыкновенных дифференциальных уравнений (Ordinary Differential Equations - ODE) методами Эйлера и Рунге-Кутты (см. Приложение Б).

4.3. В соответствии с вариантом задания (см. табл.1) построить схему моделирования линейной системы автоматического управления, используя уравнения (1-3).

Например, система 3-го порядка описывается соответствующим дифференциальным уравнением:

+=

Для решения в пакете МатЛаб преобразуем его в систему из трех дифференциальных уравнений 1-го порядка:

Тогда решение уравнения находится в виде вектора-столбца:

4.4. Создать на жестком диске директорию по собственным именем.

Активировать пакет МатЛаб. Выбрать вкладку New Script. Набрать скрипт следующего вида (в зависимости от варианта):

function [dx]=ode(t,x)

a0=…

a1=…

a2=…

a3=…

b=…

y=… / либо 1, либо sin(t) /

dx=zeros(3,1) / обнуление массива dx[3,1] /

dx(1)=x(2)

dx(2)=x(3)

dx(3)=(b\*y-a0\*x(1)-a1\*x(2)-a2\*x(3))/a3

end

Сохранить как файл **в** **своей папке** на рабочем столе, например, под именем «ode.m».

4.5 Набрать еще один скрипт следующего вида:

[t,x]=ode45(‘ode’,[0 10],[0.1 0.5 0])

/ ‘ode’ – имя 1-го скрипта

/ [0 10] – интервал времени расчета функции

/ [0.1 0.5 0] – начальные условия (в данном случае, ненулевые)

plot(t,x(:,1),'b-',t,x(:,2),’g—‘,t,sin(t),’LineWidth’,2)

/ на одном графике строим функцию x=x1, ее производную x2 и сигнал возмущения y=sin(t) (либо y=1)

legend(‘x\_1(t)’, ‘x\_2(t)’)

grid on

xlabel(‘t, c’)

ylabel(‘x\_i(t)’)

title(‘Реакция системы на входной сигнал y=sin(t) при начальных условиях [1 0.5 0.5]’)

Сохранить скрипт в ту же директорию, запустить на исполнение.

Для построения единичного ступенчатого воздействия можно использовать предварительную запись функции “у” либо:

y = ones(size(t)); % входной сигнал y = 1(t)

либо как математическую функцию от времени, тождественно равную 1, а затем оператором “plot” построить график функции “у”.

4.6. Осуществить моделирование системы с нулевыми начальными условиями при двух видах входных воздействий:

а) *y = 1(t) -* при этом график функции  будет являться графиком переходной характеристики , а график функции  будет являться графиком импульсной переходной функции .

б) *y =* sin*(t)*.

4.7. Осуществить моделирование свободного движения системы при тех же двух видах входных воздействий с ненулевыми начальными условиями в соответствии с вариантом - см. таблицу 2.

Для всех 4 вариантов выводить графики сигналов , , – если система 3 порядка, и *y(t)* в одном окне разным цветом с соответствующими легендами. Продолжительность интервала наблюдения выбрать самостоятельно, например, от 0 до 10.

Таблица 1 - Варианты параметров моделей

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Вариант | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| порядок модели  *n* | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 2 |
| *a0* | 9 | 5 | 5 | 8 | 7 | 15 | 7 | 2 | 1 | 25 | 30 | 0,12 |
| *a1* | 6 | 4 | 4 | 6 | 5 | 5 | 3 | 0,5 | 0,5 | 1 | 0,8 | 1 |
| *a2* | 3 | 3 | 2 | 2 | 10 | 0 | 1 | 1 | 0,5 | 1 | 1 | 1 |
| *a3* | 3 | 1 | 1 | 2 | 1 | 1 | - | - | - | - | - | - |
| *b0* | 12 | 2,5 | 7,5 | 12 | 10 | 15 | 10 | 4 | 2 | 25 | 30 | 0,1 |

При программировании в математическом пакете MatLab **дробная часть отделяется от целой точкой**, а не запятой.

Таблица 2 - Варианты начальных условий моделей

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Вариант | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Порядок модели *n* | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 2 |
| *x(0)* | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| *x(1) (0)* | 0,5 | -0,2 | -0,4 | 0,1 | -0,5 | 0,5 | 0,4 | 1 | -0,5 | 0 | 0,5 | 0 |
| *x(2) (0)* | 0 | 0,1 | 0,2 | -0,1 | 0 | 0,1 | - | - | - | - | - | - |

5. СОДЕРЖАНИЕ ОТЧЕТА

1. Цель работы.
2. Порядок выполнения работы.
3. Математическая модель динамической системы по варианту.
4. Графики переходных процессов.
5. Выводы.
6. ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

6.1. Какую техническую систему можно считать линейной ?

6.2. Что значит найти численное решение дифференциального уравнения ?

6.3. Найти передаточную функцию системы, динамика которой описывается дифференциальным уравнением 6*y(3)+*4*y(2)+y(1)+3y=2u(1)+5u.* Представить это уравнение в пространстве состояний и в матричном виде. Реализовать те же операции для дифференциального уравнения, сформированного по вариантам.

6.4. Метод интегрирования реализуется функцией ode45, что означает 4 и 5, каким образом гарантируется заданная точность решения ?

6.5. Показать на графике (посчитать вручную) как, зная начальные условия х(0) и у(0), получить следующую точку х(1) и у(1) методом Эйлера, модифицированным методом Эйлера, методом Рунге-Кутты 4 порядка.

**Приложение А**

**СИСТЕМА MatLAB**

Настоящее приложение содержит описание некоторых функций системы MatLAB, позволяющих моделировать технические системы и получать их характеристики в наглядном виде.

MatLAB (сокращение от Matrix Laboratory – матричная лаборатория) – это компьютерная система проведения математических расчетов, получившая широкое распространение.

Интегрирование системы обыкновенных дифференциальных уравнений

Интегрирование системы обыкновенных дифференциальных уравнений осуществляется при помощи функций ode23 и ode45. Функция ode23 осуществляет интегрирование численным методом Рунге-Кутта 2-го порядка, а с помощью метода 3-го порядка контролирует относительные и абсолютные ошибки интегрирования на каждом шаге и изменяет величину шага интегрирования так, чтобы обеспечить заданные пределы ошибок интегрирования. При использовании функции ode45 интегрирование осуществляется методом Рунге-Кутта 4-го порядка, а величина шага контролируется методом 5-го порядка.

Система дифференциальных уравнений должна быть представлена в форме Коши:

 (1.1)

где *y –* вектор переменных состояния системы, *t* – аргумент (обычно время), *f*– нелинейная вектор-функция от переменных состояния *у* и аргумента *t*.

Обращение к процедурам численного интегрирования имеет вид:

[t, y] = ode23(‘<имя функции>’, tspan, y0, options)

[t, y] = ode45(‘<имя функции>’, tspan, y0, options),

где <имя функции> - имя M-файла, являющегося функцией Matlab от *t* и *y,* в котором вычисляется вектор функция *f(y,t),* т.е. правые части системы дифференциальных уравнений;

tspan – вектор задающий интервал интегрирования [t0 tfinal], t0 – начальное значение интервала, tfinal – конечное;

yo – вектор начальных условий;

options – строка параметров, определяющих значения допустимой относительной и абсолютной погрешности интегрирования. Этот параметр можно не указывать, если пользователя устраивают значения погрешностей, заданных по умолчанию, т.е. относительная погрешность интегрирования *1.0e-3,* а абсолютная (по каждой из переменных состояния) – *1.0e-6.* В противном случае, перед обращением к процедуре ode23 следует указать значения погрешностей при помощи процедуры odeset.

Результатом интегрирования является матрица проинтегрированных значений фазовых переменных y, в которой каждый столбец соответствует одной из переменных состояния, а строка содержит значения переменных состояния, соответствующих определенному шагу интегрирования, т.е. значению вектора t.

Рассмотрим пример:

Пусть задана система обыкновенных дифференциальных уравнений:

со следующими начальными условиями:

Для интегрирования данной системы уравнений необходимо создать М-файл, который является функцией переменных *t* и *y*. Для создания файла воспользуемся редактором MATLAB Editor/Debugger, который вызывается из основного меню File – New – M-File. Текст файла:

function dy=rigid(t,y)

dy=zeros(3,1);

dy (1)=y(2)\*y(3);

dy(2)=-y(1)\*y(3);

dy(3)=-0.51\*y(1)\*y(2);

Название файла и функции должны совпадать. Файл надо сохранить с названием rigid.

В этом примере абсолютная и относительная погрешность задается при помощи команды odeset, время интегрирования зададим в интервале от 0 до 12 [0 12], вектор начальных условий [0 1 1]. Для осуществления процедуры интегрирования в рабочем пространстве Matlab необходимо набрать:

» options=odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4 1e-5]);

» [t,y]=ode45('rigid',[0 12],[0 1 1],options);

Чтобы просмотреть результаты в рабочем пространстве Matlab необходимо ввести в командной строке y. Графически результаты выводятся при помощи команды plot:

» plot(t,y(:,1),'-',t,y(:,2),'-.',t,y(:,3),'.').

Синтаксис:

            plot(y)  
            plot(x, y)  
            plot(x, y, s)  
            plot(x1, y1, s1, x2, y2, s2, ...)

Описание:

Команда plot(y) строит график элементов одномерного массива y в зависимости от номера элемента; если элементы массива y комплексные, то строится график plot(real(y), imag(y)). Если Y - двумерный действительный массив, то строятся графики для столбцов; в случае комплексных элементов их мнимые части игнорируются.

Команда plot(x, y) соответствует построению обычной функции, когда одномерный массив x соответствует значениям аргумента, а одномерный массив y - значениям функции. Когда один из массивов X или Y либо оба двумерные, реализуются следующие построения:

* если массив Y двумерный, а массив x одномерный, то строятся графики для столбцов массива Y в зависимости от элементов вектора x;
* если двумерным является массив X, а массив y одномерный, то строятся графики столбцов массива X в зависимости от элементов вектора y;
* если оба массива X и Y двумерные, то строятся зависимости столбцов массива Y от столбцов массива X.

Команда plot(x, y, s) позволяет выделить график функции, указав способ отображения линии, способ отображения точек, цвет линий и точек с помощью строковой переменной s, которая может включать до трех символов из следующей таблицы:

|  |  |  |
| --- | --- | --- |
| Тип линии | Тип точки | Цвет |
| |  |  | | --- | --- | | Непрерывная | - | | Штриховая | -- | | Двойной пунктир | : | | Штрих-пунктирная | -. | | |  |  | | --- | --- | | Точка | . | | Плюс | + | | Звездочка | \* | | Кружок | o | | Крестик | х | | |  |  | | --- | --- | | Желтый | y | | Фиолетовый | m | | Голубой | c | | Красный | r | | Зеленый | g | | Синий | b | | Белый | w | | Черный | k | |

Если цвет линии не указан, он выбирается по умолчанию из шести первых цветов, от желтого до синего, повторяясь циклически.

Команда plot(x1, y1, s1, x2, y2, s2, ...) позволяет объединить на одном графике несколько функций y1(x1), y2(x2), ..., определив для каждой из них свой способ отображения.

Обращение к командам plot вида plot(x, y, s1, x, y, s2) позволяет для графика y(x) определить дополнительные свойства, для указания которых применения одной строковой переменной s1 недостаточно, например при задании разных цветов для линии и для точек на ней.

**Примеры:**

Построим график функции y = sin(x) на отрезке [-  ] с шагом  /500:

x = -pi:pi/500:pi;  
y = sin(x);  
plot(y)  
plot(x, y)

Для вывода в одно окно нескольких графиков необходимо ввести команду hold on, для отмены — команду hold off.

Для создания нового графического окна необходимо ввести команду figure(i), где i — номер графического окна. Эта же команда используется для активизации уже созданного окна с номером i.

Для введения комментария к графикам можно воспользоваться командой legend.

Синтаксис:

             legend(‘<текст1>‘, ‘<текст2>‘, ‘<текст3>‘, ...)  
             legend(‘<тип линии1>‘, ‘<текст1>‘, ‘<тип линии2>‘, ‘<текст2>‘, ...)  
             legend(h,...)  
             legend(M)  
             legend(h, M)  
             legend off   
             legend(..., n)

Описание:

Команда legend(‘<текст1>‘, ‘<текст2>‘, ‘<текст3>‘, ...) добавляет к текущему графику пояснение в виде указанных текстовых строк.

Команда legend(‘<тип линии1>‘, ‘<текст1>‘, ‘<тип линии2>‘, ‘<текст2>‘, ...) позволяет специфицировать тип линии, которая выносится в пояснение, так, как это делается в команде plot.

Команда legend(h, ...) добавляет пояснение к графику с дескриптором h.

Команды legend(M) и legend(h, M), где M - массив строк, также допустимы для формирования пояснения. Следует помнить, что строки массива M должны иметь одинаковую длину.

Команда legend off удаляет пояснение с текущего графика.

Команда legend(..., n) устанавливает предельное количество позиций для размещения пояснения. Если оказывается, что в области графика места недостаточно, график перестраивается и пояснение размещается вне пределов графика. Если n = -1, то пояснение размещается вне области графика. Если n = 0, то пояснение размещается в области графика, если места для этого достаточно.

Для перемещения пояснения следует нажать левую кнопку мыши, находясь в этой области, а затем переместить пояснение в нужную позицию.

# ***Нахождение корней полиномов***

Система Matlab имеет функцию **roots(P)**, которая вычисляет вектор, элементы которого являются корнями заданного полинома Р.

Рассмотрим пример. Пусть задан полином:

В системе Matlab полином задается вектором его коэффициентов:

» p=[1,8,31,80,94,20]

При вводе функции roots(p) вычисляются корни полинома p:

» roots(p)

Исследование линейных стационарных систем

Исследование и ввод моделей линейных стационарных систем производится при помощи пакета системы Matlab – Control Toolbox.

*Ввод моделей в виде пространства состояний*

Рассмотрим ввод модели системы в виде пространства состояния по заданным матрицам A,B,C,D уравнений состояния системы:

 (1.2)

Матрицы вводятся в рабочем пространстве Matlab в квадратных скобках по строкам через точку с запятой, например матрица



вводится следующим образом:

» A=[0 1;-10 1]

Модель в виде пространства состояний вводится при помощи функции sys=**ss(A,B,C,D)**, где sys – произвольное название системы. Перед вводом этой команды необходимо ввести в рабочее пространство Matlab последовательно матрицы A,B,C,D.

*Ввод моделей в виде вход-выход (передаточных функций)*

Ввод модели системы в виде передаточной функции рассматривается на примере апериодического звена.

# Пусть требуется ввести модель с передаточной функцией

Для этого нужно воспользоваться функцией **tf** и в рабочем окне системы ввести данную передаточную функцию при помощи набора следующей команды:

W =**tf** ([k], [T 1])

где W- произвольное имя функции, в первой квадратной скобке вводятся коэффициенты полинома числителя (k), а во второй коэффициенты полинома знаменателя (T,1).

Рассмотрим пример со следующими коэффициентами:

*k = 10*

*T1 = 0.1*

» W=tf([10],[0.1 1])

Получение характеристик систем

#### *Расчет полюсов системы* производится при помощи команды

pole(W).

#### *Расчет нулей системы* производится при помощи команды

zero(W).

Для нахождения временных откликов системы используются функции:

##### Импульсная переходная функция ИПФ

impulse(W) – нахождение реакции системы sys на единичное импульсное входное воздействие;

## *Переходной процесс системы*

step(W) – нахождение реакции системы sys на единичное ступенчатое воздействие .

*Амплитудно-фазовую характеристику системы* в полярных координатах можно получить воспользовавшись командой

nyquist(W).

*Логарифмическую амплитудно-фазовую характеристику* системы в полярных координатах можно получить воспользовавшись командой

bode(W).

Для того чтобы построить переходной процесс системы, т.е. ее реакцию на единичное ступенчатое воздействие, а также ее частотные характеристики в одном окне используется так называемый интерактивный наблюдатель **ltiview** (для этого нужно набрать в рабочем окне команду ltiview и на экране появится окно интерактивного обозревателя). При первом обращении к обозревателю окно пусто, т.к. нужно импортировать в него модель системы.

Для этого из верхнем меню File необходимо выбрать команду import – на экране появится меню выбора импортируемой модели системы (например W).

Обозреватель позволяет получить на одном экране несколько графиков, в том числе и частотные характеристики системы. Для выбора необходимых характеристик требуется выбрать из меню Tools команду Viewer Configuration.

На экране появятся различные конфигурации количества отображаемых графиков. Если выбрать нажатием радио-кнопки конфигурацию, содержащую 4 графика, тогда на экране появятся следующие графики:

* переходной процесс;
* импульсная переходная функция (реакция системы на дельта-функцию);
* логарифмическая амплитудно-фазовая частотная характеристика;
* амплитудно-фазовая частотная характеристика в полярных координатах.

**Приложение Б**

# **Метод Эйлера. Усовершенствованный метод Эйлера. Классический метод Рунге-Кутты**

Перечисленные в заголовке методы предназначены для приближённого нахождения решений [дифференциальных уравнений](http://mathprofi.ru/differencialnye_uravnenija_primery_reshenii.html), [систем ДУ](http://mathprofi.ru/sistemy_differencialnyh_uravnenij.html), краткая постановка наиболее распространённой задачи такова:

Рассмотрим [дифференциальное уравнение первого порядка](http://mathprofi.ru/differencialnye_uravnenija_primery_reshenii.html) , для которого требуется найти частное решение, соответствующее начальному условию . Что это значит? Это значит, нам нужно найти [функцию](http://mathprofi.ru/mnozhestva.html)  (предполагается её существование), которая удовлетворяет данному дифф. уравнению, и график которой проходит через точку .

Но переменные в уравнении  разделить невозможно. Никакими известными способами. А если и возможно, то получается неберущийся интеграл. Однако частное решение существует. Здесь на помощь приходят методы приближенных вычислений, которые позволяют с высокой точностью «сымитировать» функцию   на некотором промежутке.

Идея методов Эйлера и Рунге-Кутты состоит в том, чтобы заменить фрагмент графика  ломаной линией. Рассмотрим Исторически первого и самый простой метод:

Задание

Найти частное решение дифференциального уравнения , соответствующее начальному условию , методом Эйлера на отрезке  с шагом . Построить таблицу и график приближённого решения.

Во-первых, перед нами обычное [линейное уравнение](http://mathprofi.ru/lineinye_differencialnye_uravnenija.html), которое можно решить стандартными способами, и поэтому можно сразу же найти точное решение:

 – можно выполнить проверку и убедиться, что данная функция удовлетворяет начальному условию  и является корнем уравнения .

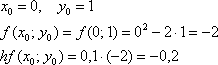
Что нужно сделать? Нужно найти и построить ломаную, которая приближает график функции  на промежутке . Поскольку длина этого промежутка равна единице, а шаг составляет , то наша ломаная будет состоять из 10 отрезков:  

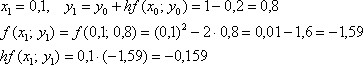

причём, точка  уже известна – она соответствует начальному условию . Кроме того, очевидны «иксовые» координаты  других точек:  

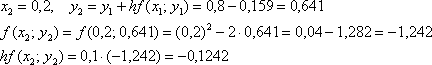

Осталось найти . Никакого [дифференцирования](http://mathprofi.ru/kak_naiti_proizvodnuju.html) и [интегрирования](http://mathprofi.ru/chto_takoe_integral_teorija_dlja_chainikov.html) – только сложение и умножение. Каждое следующее «игрековое» значение получается из предыдущего по простой рекуррентной формуле:  


Представим дифференциальное уравнение  в виде :  

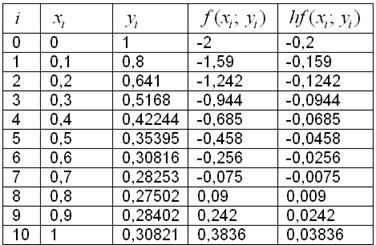

Таким образом: 

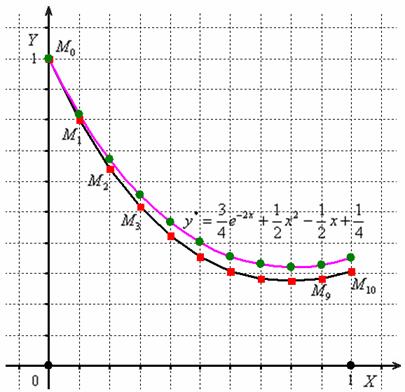
«Раскручиваемся» от начального условия :  


Далее:  




  
и так далее.

Результаты вычислений удобно заносить в таблицу:  


По результатам 2-го и 3-го столбцов изобразим на чертеже 11 точек  и 10 отрезков, соединяющих смежные точки.  Для сравнения построим график точного частного решения :  
  
Существенным недостатком простого метода Эйлера является слишком большая погрешность, при этом легко заметить, что погрешность имеет тенденцию накапливаться – чем дальше мы уходим от точки , тем преимущественно больше становится расхождение между приближением и истиной. Это объяснимо самим принципом, который Эйлер положил в основу своего метода: отрезки  параллельны соответствующим [касательным](http://mathprofi.ru/opredelenie_proizvodnoi_smysl_proizvodnoi.html) к графику функции  в точках . Данный факт хорошо просматривается по чертежу.

Как можно улучшить приближение? Первая мысль – измельчить разбиение. Разделим отрезок , например, на 20 частей. Тогда шаг составит: , и совершенно понятно, что ломаная из 20 звеньев заметно точнее приблизит частное решение. С помощью того же Экселя не составит труда обработать промежуточные отрезки, однако зададимся вопросом: а нельзя ли КАЧЕСТВЕННО улучшить метод?

## **Усовершенствованный метод Эйлера**

Рассмотрим тот же самый пример: дифференциальное уравнение , частное решение, удовлетворяющее условию , промежуток  и его разбиение на 10 частей  
( – длина каждой части).

Цель усовершенствования состоит в том, чтобы приблизить «красные квадратики» ломаной  к соответствующим «зелёным точкам» точного решения .

И идея модификации такова: отрезки  должны быть параллельны касательным, которые проведены к графику функции  не на левых краях, а «посерединке» интервалов разбиения. Что, естественно, улучшит качество приближения.

Алгоритм решения работает в том же русле, но формула, как нетрудно догадаться, усложняется:  
, где 

Плясать вновь начинаем от частного решения  и сразу же находим 1-й аргумент «внешней» функции:  


Далее следуют уже знакомые по предыдущему параграфу вычисления , после чего можно рассчитать 2-й аргумент «внешней» функции: .

Теперь находим искомую функцию , вычисленную в другой точке:  
  
Умножаем результат на шаг разбиения:  


Таким образом: 

Алгоритм заходит на второй круг:

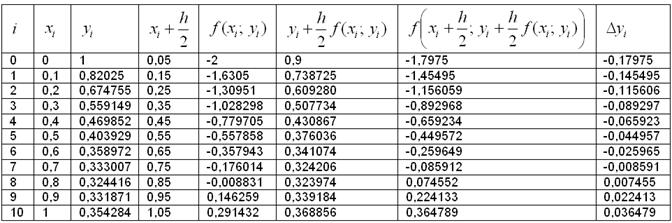
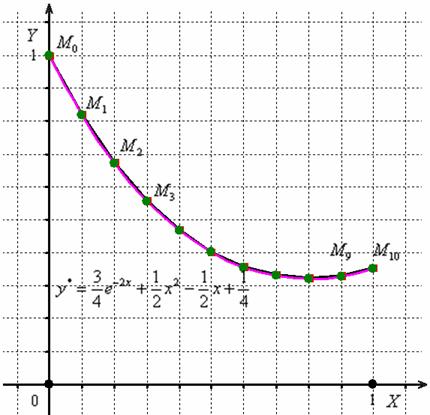
рассматриваем пару  и находим 1-й аргумент «внешней» функции:  


Рассчитываем  и находим её 2-й аргумент: 

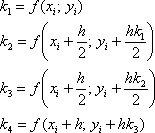
Вычислим значение:  
  
и его произведение на шаг:  

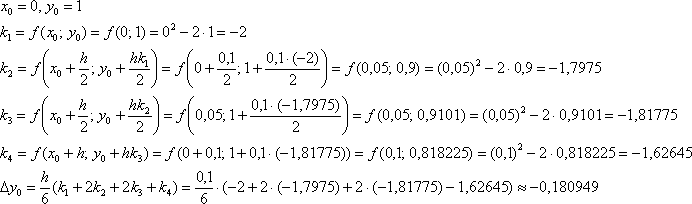

Таким образом: 

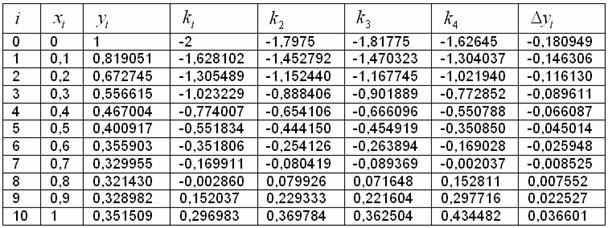
Далее рассматриваем пару  и т.д.

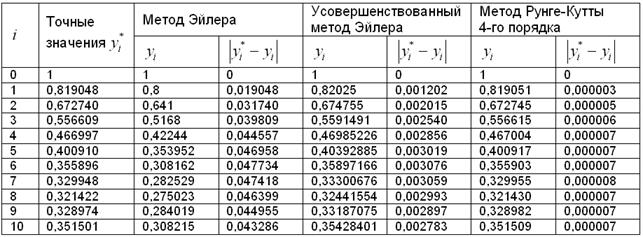
Результаты в таблице:  
  
По результатам 2-го и 3-го столбцов (слева) построим ломаную , и для сравнения график точного решения :  
  
Результат существенно улучшился – красные квадратики  практически совпали с зелёными точками точного решения.

## **Классический метод Рунге-Кутты 4-го порядка**

Его цель добиться ещё бОльшего приближения «красных квадратиков» к «зелёным точкам». Во многих, в частности физических, исследованиях бывает ПРИНЦИПИАЛЬНО важен 10-й, а то и 50-й точный знак после запятой. Такой точности можно достичь и простым методом Эйлера, но на СКОЛЬКО частей придётся разбить промежуток ?! …Хотя с современными вычислительными мощностями это не проблема. И, как правильно подсказывает заголовок, при использовании метода Рунге-Кутты на каждом шаге нам придётся вычислить значение функции  4 раза (в отличие от двукратного вычисления в предыдущем параграфе). Но задача эта вполне подъёмная. Каждое следующее «игрековое» значение получается из предыдущего  – формулы:  
, где , где:  


Активируем алгоритм:  
  
Таким образом:  


Первая строка запрограммирована, копируем формулы по образцу:  
  
В чертеже нет смысла, поскольку он уже не показателен. Проведём аналитическое сравнение точности трёх методов, т.к. известно точное решение  . Значения функции  в узловых точках элементарно рассчитываются в том же Экселе – один раз забиваем формулу  и тиражируем её на остальные .

В нижеследующую таблицу сводим значения  (для каждого из трёх методов) и соответствующие абсолютные погрешности  приближённых вычислений:  
  
Видно, что метод Рунге-Кутты даёт уже 4-5 верных знака после запятой по сравнению с 2 верными знаками усовершенствованного метода Эйлера – погрешность «обычного» метода Эйлера не превосходит шага разбиения.   
Усовершенствованный метод Эйлера гарантирует точность:  (смотрим на 2 нуля после запятой в средней колонке погрешностей).

– И, наконец, классический метод Рунге-Кутты обеспечивает точность .

Как можно ЕЩЁ улучшить точность приближения? Качеством и/или количеством. В частности, существует и другие, более точные модификации метода Рунге-Кутты. Количественный путь, как уже отмечалось, состоит в уменьшении шага, т.е. в разбиении отрезка  на бОльшее количество  промежуточных отрезков. И с увеличением этого количества ломаная  всё больше и больше будет походить на график точного решения  и в пределе – совпадёт с ним.

В математике это свойство называется спрямляемостью кривой.