**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра КСУ**

отчет

**по идз №1**

**по дисциплине «ПРОЕКТИРОВАНИЕ**

**ОПТИМАЛЬНЫХ СИСТЕМ УПРАВЛЕНИЯ»**

Тема: Решение дифференциальных уравнений

**Вариант 12**

|  |  |  |
| --- | --- | --- |
| Студент гр. 4491 | Пономарев Д.А. |  |
| Преподаватель | Ветчинкин А.С. |  |

Санкт-Петербург

2018

**Решение дифференциальных уравнений**

**Исходные данные**

При выполнении задания необходимо найти аналитическое решение уравнения путем ручных преобразований, выполнить численное решение, построить графики переходных процессов, соответствующие аналитическому и численному решениям, оценить время переходного процесса и оценить влияние выбора шага интегрирования или настроек параметров точности солверов. Для численного решения уравнений требуется разработка MATLAB скриптов, в которых реализуется метод Эйлера и применяются солверы ODE45 и ODE23S.

Исходные данные заданы в таблице 1.

Таблица 1. Исходные данные к заданию

|  |  |
| --- | --- |
| Вариант | Матрица уравнения |
| 12 |  |

**Аналитическое решение**

1. Запишем исходную систему уравнений:
2. Выполним преобразование по Лапласу и приведем к виду:
3. Воспользуемся методом Крамера:

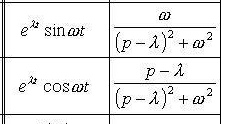
Тогда:

4. Приведем к к табличному виду.

Поскольку порядок 3 и есть комплексные корни, будем раскладывать на следующие дроби:

где I – то номер переменной состояния, – корень ХП, w – частота периодической составляющей.

В таком виде удобно будет воспользоваться таблицей преобразования Лапласа.



Пример расчета для :

Введем следующие обозначения

Тогда:

Получаем систему уравнений:

1.000\* + 0.000\* + 1.000\* = 0.000

1.327\* + 2.380\* + 1.346\*= -3.000

0.440\* + 1.556\* + 6.120\*= -4.000

Методом Крамера получаем коэффициенты:

= 0.360, = -1.257, = -0.360

В результате расчетов были получены следующие дроби:

Выполняем обратное преобразование по таблице:

Расчет времени переходного процесса

Результаты построения графиков функции представлены на рисунке 1.

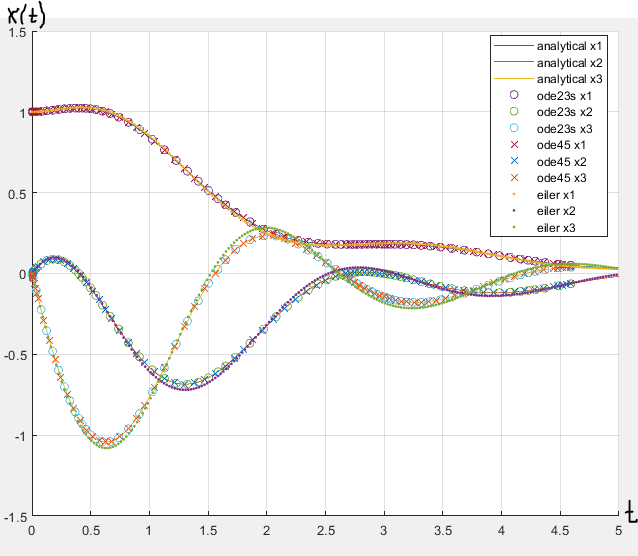


Рисунок 1 – Графики функции

На графике численного решения методом Эйлера шаг интегрирования выбран 0.005 сек.

Код программы, строящей графики:

|  |
| --- |
| clear; close all; clc;    % Input data variant12  A = [0 1 0; 1 0 2; -3 -4 -2];  B = [1 0 0];  T = 4.59; % Transient response time  % Analytical plots  figure; grid on; hold on;  lambda1 = -0.6732;  lambda2 = -0.6536;  w = 2.3805;  rootsABC = [-0.256356, 0.272450, 1.256356]  plot(t, rootsABC(3)\*exp(t\*lambda2) + rootsABC(1)\*cos(w\*t).\*exp(lambda1\*t) + rootsABC(2)\*sin(w\*t).\*exp(lambda1\*t))  rootsABC = [0.821148, 0.426843, -0.821148]  plot(t, rootsABC(3)\*exp(t\*lambda2) + rootsABC(1)\*cos(w\*t).\*exp(lambda1\*t) + rootsABC(2)\*sin(w\*t).\*exp(lambda1\*t))  rootsABC = [0.359829, -1.257274, -0.359829]  plot(t, rootsABC(3)\*exp(t\*lambda2) + rootsABC(1)\*cos(w\*t).\*exp(lambda1\*t) + rootsABC(2)\*sin(w\*t).\*exp(lambda1\*t))  % Create plot ode45, ode23s  [t, x] = ode23s('idz1\_fun', [0 T], B);  plot(t, x, 'o')  [t, x] = ode45('idz1\_fun', [0 T], B);  plot(t, x, 'x')    % Create plot Eiler  x1\_graph = []; x2\_graph = []; x3\_graph = []; t\_graph = [];  x1 = 1; x2 = 0; x3 = 0; t = 0;  dt = 0.005\*T;  while t < 5  x1\_graph = [x1\_graph x1];  x2\_graph = [x2\_graph x2];  x3\_graph = [x3\_graph x3];  t\_graph = [t\_graph t];  dx1 = (A(1, 1)\*x1 + A(1,2)\*x2 + A(1,3)\*x3)\*dt;  dx2 = (A(2, 1)\*x1 + A(2,2)\*x2 + A(2,3)\*x3)\*dt;  dx3 = (A(3, 1)\*x1 + A(3,2)\*x2 + A(3,3)\*x3)\*dt;  x1 = x1 + dx1;  x2 = x2 + dx2;  x3 = x3 + dx3;  t = t + dt;  end  plot(t\_graph, x1\_graph, '.' , ...  t\_graph, x2\_graph, '.' , ...  t\_graph, x3\_graph, '.')    % Other options  legend('analytical x1', 'analytical x2', 'analytical x3', ...  'ode23s x1', 'ode23s x2', 'ode23s x3', ...  'ode45 x1', 'ode45 x2', 'ode45 x3', ...  'eiler x1', 'eiler x2', 'eiler x3') |

|  |
| --- |
| function dxdt = idz1\_fun(t, x)  A = [0 1 0; 1 0 2; -3 -4 -2];  dxdt = [A(1, 1)\*x(1) + A(1,2)\*x(2) + A(1,3)\*x(3);...  A(2, 1)\*x(1) + A(2,2)\*x(2) + A(2,3)\*x(3);...  A(3, 1)\*x(1) + A(3,2)\*x(2) + A(3,3)\*x(3)];  end |

**Анализ влияния шага интегрирования и настроек солверов**

1. Анализ влияния настроек солверов на примере ode45

Построим 3 графика с разными настройками и выполним расчеты абсолютной ошибки, которая будет оценочным критерием качества графика. Вторым оценочным критерием будет время построения графика.

Код, дополняющий программу, представлен на рисунке 2.

|  |
| --- |
| Основной код  …  …  …  % Analysis of the influence of solver config  figure;  create\_analytical\_function\_graph();  options = odeset();  create\_graph\_for\_options\_and\_show\_analysis(options);  title("default")    figure;  create\_analytical\_function\_graph();  create\_graph\_for\_options\_and\_show\_analysis(odeset('AbsTol', 1e-7, 'RelTol', 1e-7, 'MaxStep',1e+0));  title("'AbsTol', 1e-7, 'RelTol', 1e-7, 'MaxStep',1e+0")  figure;  create\_analytical\_function\_graph();  create\_graph\_for\_options\_and\_show\_analysis(odeset('AbsTol', 1e-1, 'RelTol', 1e-1, 'MaxStep',1e+0));  title("'AbsTol', 1e-1, 'RelTol', 1e-1, 'MaxStep',1e+0")    figure;  create\_analytical\_function\_graph();  create\_graph\_for\_options\_and\_show\_analysis(odeset('AbsTol', 1e+1, 'RelTol', 1e+1, 'MaxStep',1e+0));  title("'AbsTol', 1e+1, 'RelTol', 1e+1, 'MaxStep',1e+0") |
| function create\_analytical\_function\_graph()  %@brief Create graph for analytical function in idz1  T = 4.59;  t = 0:0.01:T;  for funcCount = 1:3  funcValues = analytical\_function(t);  hold on;  plot(t, funcValues(funcCount, :), '-')  end  end |
| function x = analytical\_function(t)  %@brief calculate function values x(t)  %@param t - vector with size (n, 1)  %@return x - matrix with size (n, 3)    lambda1 = -0.6732;  lambda2 = -0.6536;  w = 2.3805;  rootsABC\_1 = [-0.2564, 0.2725, 1.2564];  rootsABC\_2 = [0.8211, 0.4268, -0.8211];  rootsABC\_3 = [0.3598, -1.2573, -0.3598];    x1 = rootsABC\_1(3).\*exp(t.\*lambda2) + rootsABC\_1(1).\*cos(w.\*t).\*exp(lambda1.\*t) + rootsABC\_1(2).\*sin(w\*t).\*exp(lambda1.\*t);  x2 = rootsABC\_2(3).\*exp(t.\*lambda2) + rootsABC\_2(1).\*cos(w.\*t).\*exp(lambda1.\*t) + rootsABC\_2(2).\*sin(w\*t).\*exp(lambda1.\*t);  x3 = rootsABC\_3(3).\*exp(t.\*lambda2) + rootsABC\_3(1).\*cos(w.\*t).\*exp(lambda1.\*t) + rootsABC\_3(2).\*sin(w\*t).\*exp(lambda1.\*t);  x = [x1; x2; x3];  end |
| function create\_graph\_for\_options\_and\_show\_analysis(opt)  %@brief Create graph ode45 and show analysis:  %- tic/toc  %- analysis (see do\_analysis())  %@param opt - ode45 options object    global T B    tic  [t, x] = ode45('idz1\_fun', [0 T], B, opt);  toc  grid on; hold on;  plot(t, x, 'o')    legend('x1(t) analytical', ...  'x2(t) analytical', ...  'x3(t) analytical', ...  'x1(t) ode45', ...  'x2(t) ode45', ...  'x3(t) ode45')  xlabel('t'); ylabel('x(t)');    do\_analysis(t, x)  end |
| function do\_analysis(t, x)  %@brief Find and show:  %-max absolution error  %-regulation time  %@param t - array of time with size = (n, 1)  %@param x - value of function with size = (n, 3)    % 1. Find max absolution error  maxAbsErrorValues = zeros(1, 3);  maxAbsErrorTimes = zeros(1, 3);  for i = 1:length(t)  analyticalValues = analytical\_function(t(i));  calculatedValues = (x(i, :))';  errors = analyticalValues - calculatedValues;  for funcCount = 1:3  if abs(errors(funcCount)) > maxAbsErrorValues(funcCount)  maxAbsErrorValues(funcCount) = abs(errors(funcCount));  maxAbsErrorTimes(funcCount) = t(i);  end  end  end  for funcCount = 1:3  fprintf('Max absolute error x%d on %f sec = %f\n', funcCount, maxAbsErrorTimes(funcCount), maxAbsErrorValues(funcCount))  end    % 2. Find regulation time  regulationTime = zeros(1, 3);  steadyStateValueUpperBorder = 0.05;  steadyStateValueLowerBorder = -0.05;  for funcCount = 1:3  for count = length(t):-1:1  if(x(count, funcCount) > steadyStateValueUpperBorder) || (x(count, funcCount) < steadyStateValueLowerBorder)  regulationTime(funcCount) = t(count);  fprintf('Regulation time x%d = %f\n', funcCount, regulationTime(funcCount))  break  end  end  end |

Рисунок 2 – дополнение к коду программы

Результат выполнения программы представлен на рисунке 3.

|  |
| --- |
| Elapsed time is 0.020295 seconds.  Max absolute error x1 on 1.130274 sec = 0.000096  Max absolute error x2 on 2.667467 sec = 0.000152  Max absolute error x3 on 2.054802 sec = 0.000222  Regulation time x1 = 4.590000  Regulation time x2 = 4.590000  Regulation time x3 = 3.934278  Elapsed time is 0.036968 seconds.  Max absolute error x1 on 0.778586 sec = 0.000059  Max absolute error x2 on 1.402650 sec = 0.000043  Max absolute error x3 on 1.470930 sec = 0.000023  Regulation time x1 = 4.590000  Regulation time x2 = 4.590000  Regulation time x3 = 3.947075  Elapsed time is 0.003990 seconds.  Max absolute error x1 on 0.588894 sec = 0.016259  Max absolute error x2 on 2.051964 sec = 0.040140  Max absolute error x3 on 1.215255 sec = 0.058699  Regulation time x1 = 4.370146  Regulation time x2 = 4.370146  Regulation time x3 = 3.710584  Elapsed time is 0.005169 seconds.  Max absolute error x1 on 1.672638 sec = 0.044378  Max absolute error x2 on 2.172638 sec = 0.088121  Max absolute error x3 on 1.422638 sec = 0.139797  Regulation time x1 = 4.464479  Regulation time x2 = 4.172638  Regulation time x3 = 4.172638 |

Рисунок 3 – Результат выполнения программы

Графики функций, построенные при различных настройках солвера, представлены на рисунках 4, 5, 6, 7.

Как видно из графиков, при увеличении точности ode45 с помощью настроек абсолютной и относительной точности AbsTol и RelTol, точность действительно увеличивается. Однако при этом увеличивается и время выполнения расчетов. Также, увеличенное количество точек значительно увеличивает время построения графиков.

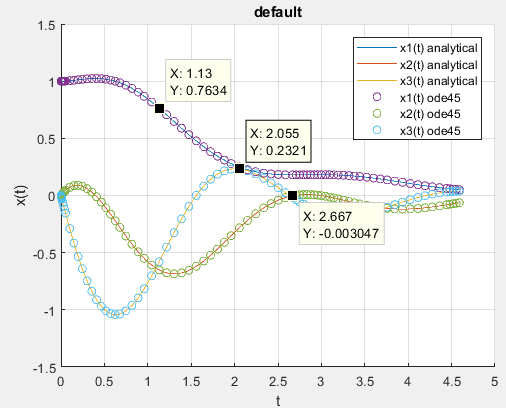


Рисунок 4 – График функции, полученный с помощью ode45 с стандартными настройками

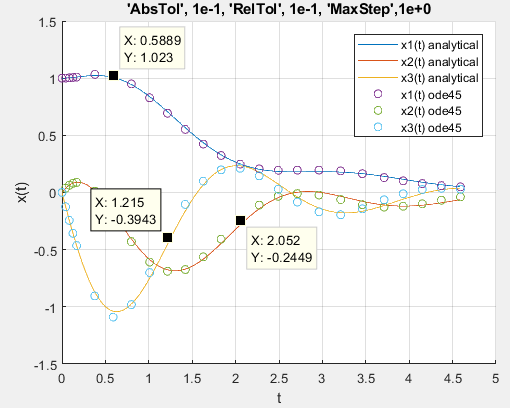


Рисунок 5 – График функции, полученный с помощью ode45 с нестандартными настройками

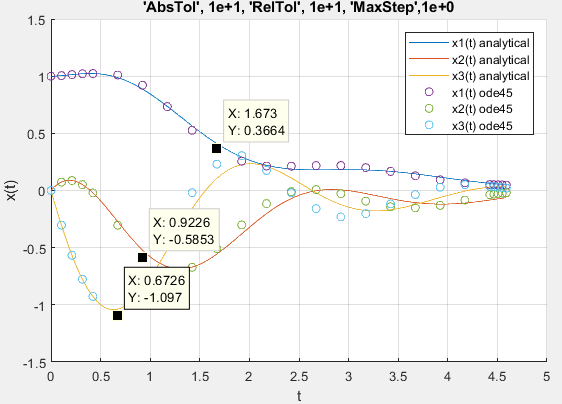


Рисунок 6 – График функции, полученный с помощью ode45 с нестандартными настройками

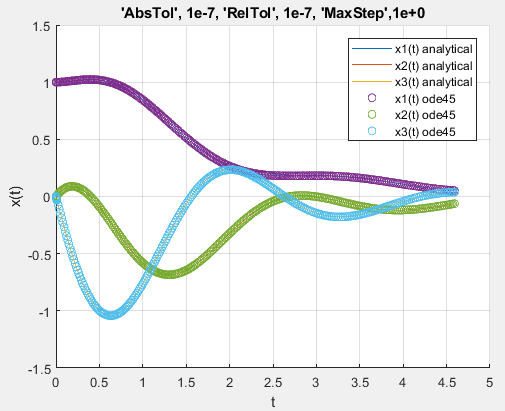


Рисунок 5 – График функции, полученный с помощью ode45 с нестандартными настройками

**2. Анализ влияния шага интегрирования**

Построим 3 графика с разными шагами интегрирования и выполним расчеты абсолютной ошибки, которая будет оценочным критерием качества графика.

Код, дополняющий программу, представлен на рисунке 6.

|  |
| --- |
| Основной код  …  …  …  %Analysis of the influence of the integration step  figure;  create\_analytical\_function\_graph()  create\_eiler\_graph\_with\_step\_and\_show\_analysis(0.001)  create\_eiler\_graph\_with\_step\_and\_show\_analysis(0.01)  create\_eiler\_graph\_with\_step\_and\_show\_analysis(0.05)  grid on; |
| function create\_eiler\_graph\_with\_step\_and\_show\_analysis(step)  %@brief Create graph with method Eiler and show analysis  %@param step - integration step (scalar)    global A T    x1\_graph = []; x2\_graph = []; x3\_graph = []; t\_graph = [];  x1 = 1; x2 = 0; x3 = 0; t = 0;  dt = step;  while t < T\*1.5  x1\_graph = [x1\_graph x1];  x2\_graph = [x2\_graph x2];  x3\_graph = [x3\_graph x3];  t\_graph = [t\_graph t];  dx1 = (A(1, 1)\*x1 + A(1,2)\*x2 + A(1,3)\*x3)\*dt;  dx2 = (A(2, 1)\*x1 + A(2,2)\*x2 + A(2,3)\*x3)\*dt;  dx3 = (A(3, 1)\*x1 + A(3,2)\*x2 + A(3,3)\*x3)\*dt;  x1 = x1 + dx1;  x2 = x2 + dx2;  x3 = x3 + dx3;  t = t + dt;  end  hold on;  plot(t\_graph, x1\_graph, '-' , ...  t\_graph, x2\_graph, '-' , ...  t\_graph, x3\_graph, '-')    fprintf("Step = %f\n", step);  do\_analysis(t\_graph', [x1\_graph; x2\_graph; x3\_graph]')  end |

Рисунок 6 – Код, дополняющий основную программу, необходимый для анализа шага интегрирования

Результат выполнения программы представлен на рисунке 7.

|  |
| --- |
| Step = 0.001000  Max absolute error x1 on 2.020000 sec = 0.000731  Max absolute error x2 on 1.306000 sec = 0.001482  Max absolute error x3 on 1.844000 sec = 0.002159  Regulation time x1 = 4.601000  Regulation time x2 = 4.691000  Regulation time x3 = 3.966000  Step = 0.010000  Max absolute error x1 on 2.030000 sec = 0.007535  Max absolute error x2 on 1.310000 sec = 0.014744  Max absolute error x3 on 1.840000 sec = 0.022323  Regulation time x1 = 4.550000  Regulation time x2 = 4.670000  Regulation time x3 = 3.950000  Step = 0.050000  Max absolute error x1 on 2.050000 sec = 0.042009  Max absolute error x2 on 2.550000 sec = 0.082824  Max absolute error x3 on 1.850000 sec = 0.125660  Regulation time x1 = 4.350000  Regulation time x2 = 4.600000  Regulation time x3 = 6.050000 |

Рисунок 7 – Результат выполнения программы

Графики представлены на рисунке 8.

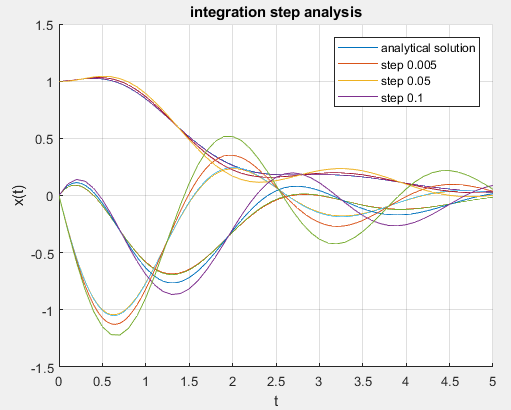


Рисунок 8 – Результат выполнения программы с разным шагом интегрирования

Вывод: как видно из графика, при увеличении шага интегрирования, увеличивается максимальная ошибка и время регулирования.