**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра КСУ**

**КУРСОВАЯ РАБОТА**

**по дисциплине «ПРОЕКТИРОВАНИЕ**

**ОПТИМАЛЬНЫХ СИСТЕМ УПРАВЛЕНИЯ»**

Тема: проектирования алгоритма управления динамическим объектом на примере водоизмещающего судна

**Вариант 12**

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

Санкт-Петербург

2018

Задание на курсовую работу

Студент: Пономарев Д.А.

Группа: 4491

Тема работы:

Исходные данные представлены в таблицах 1, 2 и 3.

Таблица 1 - Вариант курсового расчета

|  |  |  |
| --- | --- | --- |
| Вариант задания курсового расчета | Вариант параметров математической модели судна | Косвенный метод решения задачи оптимизации |
| 12 | 1 – 8 | Использование стандартного полинома П2.1 |

Таблица 2 - Вариант математической модели судна

|  |  |  |
| --- | --- | --- |
| Параметр | Обозначение | Вариант судна |
| 4 |
| Скорость хода | , м/сек | 6.17 |
| Длина по ватерлинии | , м | 39 |
| Коэффициенты математической модели |  | -0.69 |
|  | 6.14 |
|  | 1.22 |
|  | -3.12 |
|  | -0.44 |
|  | -3.1 |



Таблица 3 - П2.1 Биноминальные полиномы

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
| 1 | 1 | 1 |  |  |  |  |  |  |  |
| 2 | 1 | 2 | 1 |  |  |  |  |  |  |
| 3 | 1 | 3 | 3 | 1 |  |  |  |  |  |
| 4 | 1 | 4 | 6 | 4 | 1 |  |  |  |  |
| 5 | 1 | 5 | 10 | 10 | 5 | 1 |  |  |  |
| 6 | 1 | 6 | 15 | 20 | 15 | 6 | 1 |  |  |
| 7 | 1 | 7 | 21 | 35 | 35 | 21 | 7 | 1 |  |
| 8 | 1 | 8 | 28 | 56 | 70 | 56 | 28 | 8 | 1 |

Дача сдачи реферата:

Дата защиты реферата:

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Пономарев Д.А.

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ветчинкин А.С.

Аннотация

Курсовой расчет предназначен для ознакомления студентов с процессом проектирования алгоритма управления динамическим объектом на примере водоизмещающего судна.

Summary

Course work is intended to familiarize students with the process of designing a dynamic object control algorithm using the example of a displacement vessel.

Оглавление

[Задание на курсовую работу 2](#_Toc532234981)

[Аннотация 3](#_Toc532234982)

[Summary 3](#_Toc532234983)

[Введение 5](#_Toc532234984)

[1. Математическое описание объекта управления 6](#_Toc532234985)

[2. Математическая формулировка цели управления 8](#_Toc532234986)

[3. Выбор и реализация метода решения оптимизационной задачи 9](#_Toc532234987)

[3.1. Метод, основанный на теореме об N интервалах 9](#_Toc532234988)

[3.2. Метод параметрической оптимизации линейного закона управления 18](#_Toc532234989)

[3.3. Метод стандартного полинома - биноминальные полином. 22](#_Toc532234990)

[4. Анализ чувствительности реализованных методов 28](#_Toc532234991)

[Заключение 30](#_Toc532234992)

[Список литературы 31](#_Toc532234993)

[Приложение 32](#_Toc532234994)

Введение

Курсовой расчет предназначен для ознакомления с процессом проектирования алгоритма управления динамическим объектом на примере водоизмещающего судна.

Проектирование алгоритма управления состоит из следующих этапов:

- математическое описание объекта управления

- математическая формулировка цели управления

- выбор метода решения поставленной оптимизационной задачи

- оценка вариантов решения задачи

1. Математическое описание объекта управления

Динамика судна, как и любого физического тела, подчиняется второму закону Ньютона. Силы и моменты, действующие на судно, в свою очередь, описываются законами гидродинамики. Соотношения между кинематическими параметрами движения ( - угол рыскания,  - угловая скорость рыскания,  - угол дрейфа,  - угол перекладки руля) показаны на рисунке 1.

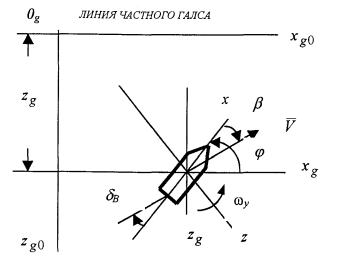


Рисунок 1 – соотношения между кинематическими параметрами

В общем случае, зависимость сил и моментов, действующих на судно от параметров движения носит нелинейный характер. Однако предположение о малых значениях угла дрейфа и угловой скорости рыскания и постоянстве линейной скорости движения судна позволяют линеаризовать эти зависимости и описать динамику в виде системы линейных дифференциальных уравнений относительно углов рыскания, дрейфа, угловой скорости рыскания, угла перекладки руля и одного нелинейного соотношения, отражающего тот факт, что руль не может поворачиваться на произвольный угол при произвольном сигнале управления. Для большинства современных судов максимальный угол перекладки руля равен 35°. Упомянутые соотношения, записанные относительно нормированного времени , имеют вид (1). При записи (1), кроме предположений о малости углов не учитывалось действие на судно ветро-волновых возмущений. т.е. математическая модель (1) соответствует движению судна на тихой воде.

 (1)

где:  - относительная скорость рыскания;  - угол дрейфа;  - угол перекладки руля.

Математическая модель судна в натуральном времени записывается в виде (2).

 (2)

Соотношение между параметрами (1) и (2) имеет вид (3).

 (3)

Значение нормирующей частоты  вычисляется по данным таблиц 1 и 2, содержащим варианты параметров математических моделей судов.

Система уравнений четвертого порядка, учитывающая ограничение на угол и скорость перекладки руля:

Аналитический вывод системы уравнений, с помощью которой могут быть рассчитаны коэффициенты k:

Характеристический полином равен:

Биномиальный полином равен:

Приравняв оба полинома, получаем систему из четырех уравнений, из которой можно получить с помощью MATLAB функции solve значения k1, k2, k3 и k4:

2. Математическая формулировка цели управления

При выполнении настоящей курсовой работы требуется спроектировать алгоритм управления рулем судна, который обеспечивает минимальное время устранения начального значения угла рыскания равного 10°.

3. Выбор и реализация метода решения оптимизационной задачи

В рамках настоящего курсового расчета выполнено проектирование алгоритма управления тремя методами:

1. два прямых метода:

1.1 метод, основанный на теореме об N интервалах

1.2 метод параметрической оптимизации линейного закона управления

2. косвенный метод:

2.2 метод стандартного полинома

3.1. Метод, основанный на теореме об N интервалах

**Описание метода**

Метод, основанный на теореме об N интервалах заключается в определении N-1 момента переключения знака управляющего воздействия и момента выключения управления, таких, которые обеспечивают перевод судна из начального состояния , ,  в конечное состояние , ,  к моменту времени выключения управления, где N - порядок дифференциального уравнения (или системы уравнений), описывающего объект управления.

Проверим применимость теоремы об N интервалов. Для этого определим корни характеристического полинома.

[ s - a11, -a21, 0]

A= [-a21, s –a22, 0]

[-1, 0, s]

det(A1)=s^3 - s^2\*(a11 + a22)+s(a11\*a22 - a21\*a12)=0

s = {0, 1.0353, 0.1290}

Поскольку характеристическое уравнение рассматриваемого объекта имеет вещественные корни, то к нему применима теорема об N интервалах, из которой следует, что в рассматриваемом случае для перевода объекта управления между заданными точками управляющее воздействие должно быть максимально по модулю и менять знак не более двух раз.

Задачу определения моментов переключения  и момента выключения  решается поисковым методом используя функцию FMINSEARCH из пакета MATLAB.

При выполнении расчетов учитывается, что поисковые методы не могут обеспечить точное решение за конечное количество шагов. В связи с этим целесообразно смягчить поставленную задачу, потребовав минимизировать отклонение состояния объекта от требуемого в момент выключения управления.

Известно, что результат, получаемый с помощью поисковых методов, может существенно зависеть от выбора начальной точки поиска. В связи с этим при проектировании алгоритма управления на основе теоремы об N интервалах выполняется мероприятие, направленное на определение начальной комбинации искомых параметров.

Поскольку уравнения по угловой скорости рыскания и углу дрейфа для рассматриваемого объекта управления не зависят от угла рыскания, то первое приближение для набора искомых параметров можно определить с помощью решения промежуточной задачи пониженного порядка.

В качестве промежуточной удобно рассмотреть задачу перевода судна из состояния ,  в состояние ,  за счет выполнения двух переключений знака угла перекладки руля.

При выборе произвольного значения для первого момента переключения, соответствующие значения для второго переключения и момента выключения могут быть определены графическим методом на основе построения линии переключения в плоскости . В результате решения промежуточной задачи определяется набор параметров , которые соответствуют переводу судна из заданного начального состояния в состояние промежуточного финиша , которое отличается от заданного конечного состояния . Определение окончательного набора параметров  может быть организовано в виде итеративной процедуры поиска минимума функции 

На каждом шаге итеративной процедуры в качестве начального набора параметров  при поиске функции минимума  используется результат предыдущего шага, а значение  систематически приближается к заданному конечному значению угла рыскания .

**Описание кода программы**

Основным скриптом является скрипт main.m. В нем последовательный вызов трех функций: расчета параметров системы, решения вспомогательной задачи и решение основной задачи.

Функция auxiliary\_task выполняет построение линии переключения в плоскости x1x2 и построение фазовой траектории объекта управления в соответствии с моментами переключения/выключения , заданными в программе.

Работа с программой для определения второго момента переключения выполняется следующим образом:

- параметр t1 задается произвольно;

- параметр ti задается равным t2 для ограничения времени построения фазовой траектории:

- параметр t2 подбирается так, чтобы соответствующая этому значеню траектория прикоснулась к линии переключения;

- величина t2, соответствующая соприкосновению фазовой траектории и линии переключения объявляется моментом второго переключения

- величины параметров t1и t2 сохряются для последующих экспериментов.

Работа с программой для определения момента выключения выполняется следующим образом:

- параметры t1 и t2 сохраняют свои значения после предыдущих экспериментов;

- параметр ti задается равным T для ограничения времени построения фазовой траектории;

- параметр T подбирается так чтобы к моменту времени t=T фазовая траектория пришла в требуемую конечную точку x1=0, x2=0.

После окончания подбора величины параметра T запоминается величина x3(T), которая объявляется точкой промежуточного финиша x3f.

После выполнения описанных экспериментов становится известной комбинация параметров , которая позволяет перевести объект управления в заданную точку плоскости x1x2. Однако эта комбинация параметров управления в пространстве  вместо заданной точки  приводит объект в точку промежуточного финиша [0,0,x3f].

Для получения результатов, сопоставимых с результатами решения задачи управления другими рассматриваемыми в курсовом расчете методами (модальное управление, параметрическая оптимизация, …) целесообразно за время переходного процесса принимать не момент попадания изображающей точки в заданное положение, а момент времени после которого длина вектора состояния становится менее 5% от начального значения.

**Код программы**

Main.m – основной файл

|  |
| --- |
| clc; clear; close all;  % Инициализация глобальных переменных системы  init\_system()  % Part 1. Вспомогательная задача. Выбираем t1 произвольно, t2 и T опреде-  % ляются графическим способом  global t1 t2 T  %4.7, 8.31, 8.958 => ~0.7  t1 = 4.915;  t2 = 8.603;  T = 9.25;  auxiliary\_task()  % Part 2. Уточнение комбинации переключающих моментов  main\_task() |

init\_system.m –

|  |
| --- |
| function init\_system()  global A B t0 x\_start x\_end  x\_start = [0 0 10];  x\_end = [0 0 0];  t0 = 0;  V0 = 6.17; L = 39; r21 = -0.69; r31 = 6.14; q21 = 1.22;  q31 = -3.12; s21 = -0.44; s31 = -3.1; W = V0/L; a11 = -r31\*W;  a12 = -q31\*W^2; a21 = -r21; a22 = -q21\*W; b11 = -s31\*W^2; b21 = -s21\*W;  A = [a11 a12 0; % угловая скорость рысканья омега\_у  a21 a22 0; % угол дрейфа бетта  1 0 0]; % угол рысканья фи  B = [b11; b21; 0]; % коэффицинты перед управляющим воздействием  end |

auxiliary\_task.m – функция для решения вспомогательной задачи

|  |
| --- |
| function auxiliary\_task()  global t0 t1 t2 t T x x\_start x\_end x3f  figure; hold on; grid on;  [t\_inverse, x\_inverse] = ode45('odefun\_with\_negative\_u', [t0 1], x\_end);  [t, x] = ode23s('odefun', [t0 T], x\_start);  subplot(3, 1, 2), plot(x(:,2), x(:,1)); title('Фазовая плоскость x1(x2)')  hold on; plot(x\_inverse(:,2), x\_inverse(:,1));  subplot(3, 1, 1), plot(t, x); title('Переходные процессы x1(t), x2(t), x3(t)')  subplot(3, 1, 3), plot(t, calculate\_u(t), 'b'); title('Управляющее воздействие u(t)')  fprintf("Initial combination of switching times are:\nt1 = %f, t2 = %f, t3 = %f\n", t1, t2, T);  fprintf("Initial end point is:\nx1\_end = %f, x2\_end = %f, x3\_end = %f\n\n", x(end, :));  x3f = x(end,3);  end |

main\_task.m - файл для задачи уточнения комбинации моментов переключения

|  |
| --- |
| function main\_task()  global t1 t2 T x t  figure; grid on;  fminsearch('fmsfun', [t1 t2 T]);  fprintf("Specify combination of switching times are:\nt1 = %f, t2 = %f, t3 = %f\n", t1, t2, T);  fprintf("Specify end point is:\nx1\_end = %f, x2\_end = %f, x3\_end = %f\n\n", x(end, :));  fprintf("Transition time = %f\n\n", calculate\_transition\_time(t, x));  end |

fmsfun.m – функция для fminsearch

|  |
| --- |
| function discrepancy = fmsfun(switchingTimesArray)  global t1 t2 T t x x\_start x\_end x3f  t1 = switchingTimesArray(1); t2 = switchingTimesArray(2) ; T = switchingTimesArray(3);  [t,x]=ode23s('odefun',[0 T], x\_start);  discrepancy = (x(end,1)-x\_end(1))^2 + (x(end,2)-x\_end(2))^2 + (x(end,3)-x3f)^2;  subplot(3, 1, 1), plot(t, x); title('Переходные процессы x1(t), x2(t), x3(t)')  subplot(3, 1, 2), plot(x(:,2), x(:,1), 'b'); title('Фазовая плоскость x1(x2)')  subplot(3, 1, 3), plot(t, calculate\_u(t), 'b'); title('Управляющее воздействие u(t)')  pause(0.1)  end |

odefun.m – функция для ode23s

|  |
| --- |
| function dxdt = odefun(t, x)  global A B  uMax = 35;  dxdt = [A(1,1)\*x(1) + A(1,2)\*x(2) + A(1,3)\*x(3) + uMax\*B(1)\*calculate\_u(t); ...  A(2,1)\*x(1) + A(2,2)\*x(2) + A(2,3)\*x(3) + uMax\*B(2)\*calculate\_u(t); ...  A(3,1)\*x(1) + A(3,2)\*x(2) + A(3,3)\*x(3) + uMax\*B(3)\*calculate\_u(t)]; ...  end |

Odefun\_negative.m – функция для ode23s

|  |
| --- |
| function dxdt = odefun\_with\_negative\_u(t, x)  global A B  u = -35;  dxdt = -[A(1,1)\*x(1) + A(1,2)\*x(2) + A(1,3)\*x(3) + B(1)\*u; ...  A(2,1)\*x(1) + A(2,2)\*x(2) + A(2,3)\*x(3) + B(2)\*u; ...  A(3,1)\*x(1) + A(3,2)\*x(2) + A(3,3)\*x(3) + B(3)\*u]; ...  end |

calculate\_transition\_time.m – функция расчета времени переходного процесса

|  |
| --- |
| function transitionTime = calculate\_transition\_time(t, x)  %@brief Calculate transition time for each state variabels and return max  %@param t - time vector with size (pointsAmount, 1)  %@param x - state variables matrix with size(pointsAmount, stateVariabelsAmount)  %@return max transition time scalar  pointsAmount = size(x, 1);  transitionTime = t(end);  for i = pointsAmount : -1 : 1  vectorLength = (x(i,1)^2 + x(i,2)^2 + x(i,3)^2)^0.5;  if vectorLength > 0.05\*10  transitionTime = t(i);  break;  end  end  end |

Файл

|  |
| --- |
| function u = calculate\_u(t)  global t1 t2 T  u = zeros(length(t), 1);  for i = 1:length(t)  if t(i) <= t1  u(i) = -1;  elseif t(i) <= t2  u(i) = 1;  elseif t(i) <= T  u(i) = -1;  else  u(i) = 0;  end  end  end |

Результат выполнения решения вспомогательной задачи представлен на рисунке 1.

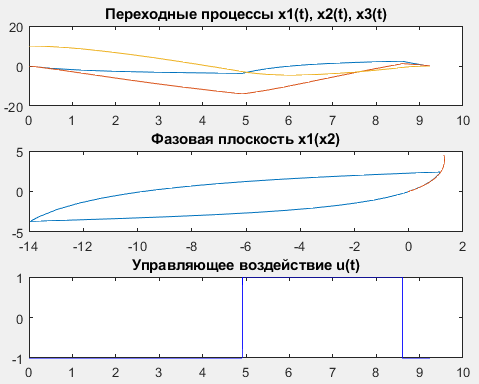


Рисунок 1 – Результаты решения вспомогательной задачи

Результаты, выведенные в командное окно:

Initial combination of switching times are:

t1 = 4.915000, t2 = 8.603000, t3 = 9.250000

Initial end point is:

x1\_end = 0.001893, x2\_end = -0.007409, x3\_end = -0.015700

Результат выполнения задачи уточнения точек переключения представлены на рисунке 2.

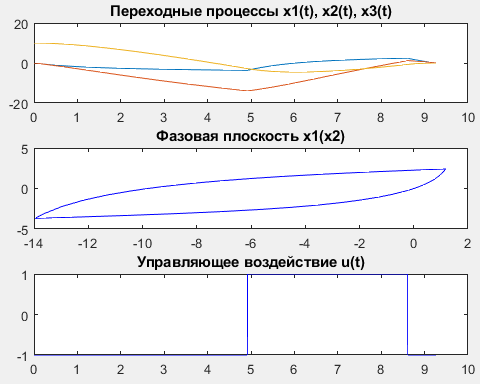


Рисунок 2 – Результаты решения задачи уточнения точек переключения

Результаты, выведенные в командное окно:

Specify combination of switching times are:

t1 = 4.917494, t2 = 8.609763, t3 = 9.257274

Specify end point is:

x1\_end = -0.000043, x2\_end = -0.000232, x3\_end = -0.016814

Таким образом, обеспечивается перевод судна из начального состояния , ,  в конечное состояние , ,  за время t = 9.257 сек.

3.2. Метод параметрической оптимизации линейного закона управления

**Описание метода**

Метод параметрической оптимизации линейного закона управления заключается в поиске таких значений параметров линейного закона управления, которые обеспечивают перевод объекта управления в заданное состояние за минимальное время и последующее удержание объекта в этом состоянии.

Одним из достоинств этого метода является возможность включения в закон управления только тех переменных состояния, которые соответствуют достаточно точно измеряемым физическим величинам. В случае водоизмещающего судна наиболее точно из принятых в рассмотрение физических величин измеряются угол рыскания и угловая скорость рыскания.

С математической точки зрения задача параметрической оптимизации заключается в том, чтобы для алгоритма управления  найти такие значения параметров  и  при которых время перевода судна из начальной точки в конечную происходит за минимальное время.

Поскольку в рассматриваемом случае изменение состояния судна вблизи целевой точки носит экспоненциальный характер, то теоретически время перехода в целевое состояние не ограничено. В связи с этим предлагается принимать за момент окончания процесса управления момент времени, после которого абсолютное значение угла рыскания не превышает 5% от начального значения

Очевидно, что для решения этой задачи удобно использовать MATLAB функцию FMINSEARCH, которая возвращает искомые значения параметров алгоритма управления в виде вектора, а в процессе поиска вызывает процедуру определения времени переходного процесса, соответствующего текущим значениям искомых параметров. В свою очередь реализация процедуры определения времени переходного процесса может быть основана на применении MATLAB функции ODE45, обеспечивающей получение решения дифференциальных уравнений замкнутой системы управления в виде массива значений . Кроме вызова функции ODE45 для определения длительности переходного процесса необходимо выполнить обработку массива значений .

**Код программы**

Код программы представлен ниже.

Файл main.m – основной файл, запускаем его

|  |
| --- |
| clc; clear; close all;  % Calculate parameters  global A B  V0 = 6.17; L = 39; r21 = -0.69; r31 = 6.14; q21 = 1.22;  q31 = -3.12; s21 = -0.44; s31 = -3.1; W = V0/L; a11 = -r31\*W;  a12 = -q31\*W^2; a21 = -r21; a22 = -q21\*W; b11 = -s31\*W^2; b21 = -s21\*W;  A = [a11 a12 0 b11; % угловая скорость рысканья омега\_у  a21 a22 0 b21; % угол дрейфа бетта  1 0 0 0; % угол рыскань¤ фи  0 0 0 0]; % угол поворота руля  B = [0; 0; 0; 1]; % коэффицинты перед управл¤ющим воздействием  % Main algorithm  parametersStart = [1.3; 10.5];  [k, time] = fminsearch('fminearch\_function', parametersStart) |

fminearch\_function.m – файл с функцией для fminsearch

|  |
| --- |
| function transitionTime = fminearch\_function(parameters)  global K  K = parameters;  % Численное решение при заданных коэффициентах К  [t, x] = ode45('odefun', [0 50], [0 0 10 0]);  transitionTime = calculate\_transition\_time(t, x);  % Построение графиков  u = zeros(length(t), 1);  for i = 1:length(t)  u(i) = control\_impact(x(i, :));  end  subplot(3, 1, 1); plot(t, x(:, 1:3)); title('Переходные процессы, фи(t), w(t), B(t)'); grid on;  subplot(3, 1, 2); plot(t, u); title('Скорость поворота руля, u(t)'); grid on;  subplot(3, 1, 3); plot(t, x(:, 4)); title('Угол поворота руля, d(t)'); grid on;  pause(0.1)  end |

Odefun – функция для ode45

|  |
| --- |
| function dxdt = odefun(t, x)  global A B  rudderAngleMax = 35;  if x(4) > rudderAngleMax  x(4) = rudderAngleMax;  elseif x(4) < -rudderAngleMax  x(4) = -rudderAngleMax;  end  dxdt = [A(1,1)\*x(1) + A(1,2)\*x(2) + A(1,3)\*x(3) + A(1,4)\*x(4); ...  A(2,1)\*x(1) + A(2,2)\*x(2) + A(2,3)\*x(3) + A(2,4)\*x(4); ...  A(3,1)\*x(1) + A(3,2)\*x(2) + A(3,3)\*x(3) + A(3,4)\*x(4);  control\_impact(x)];  end |

control\_impact – функция расчета управляющего воздействия

|  |
| --- |
| function u = control\_impact(x)  %@brief Calculate control impact u = -K1\*x1 - K2\*x2  %@param x - state variables with size (pointsAmount, stateVariablesAmount)  %@return control impact with size(pointsAmount, 1)  %@note u <= |Umax|  global K  Umax = 1;  u = -K(1).\*x(3) - K(2).\*x(1);  if u > Umax  u = Umax;  elseif u < -Umax  u = -Umax;  end  end |

calculate\_transition\_time – функция расчета времени переходного процесса

|  |
| --- |
| function transitionTime = calculate\_transition\_time(t, x)  %@brief Calculate transition time for each state variabels and return max  %@param t - time vector with size (pointsAmount, 1)  %@param x - state variables matrix with size(pointsAmount, stateVariabelsAmount)  %@return max transition time scalar  pointsAmount = size(x, 1);  transitionTime = t(end);  for i = pointsAmount : -1 : 1  if abs(x(i,3)) > 0.05\*10  transitionTime = t(i);  break;  end  end  end |

**Результат выполнения программы**

Результат выполнения программы представлен на рисунке 3.

В ходе выполнения программы, были вычислены коэффициенты управляющего воздействия:

K1 = 1.3662, K2 = 11.2614

А время переходного процесса получилось:

time =16.7530

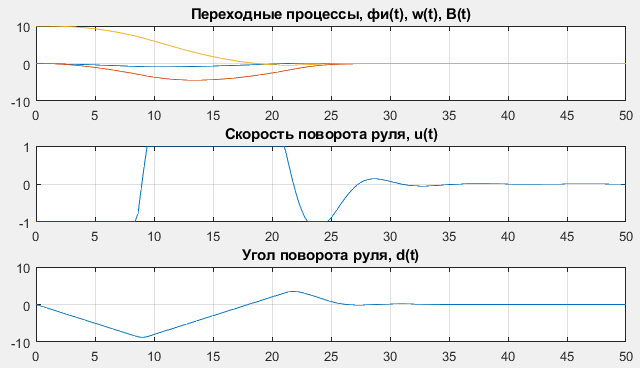


Рисунок 3 – Переходные процессы системы, полученные параметрическим методом

Таким образом, были найдены такие значения параметров линейного закона управления, которые обеспечивают перевод объекта управления в заданное состояние за минимальное время и последующее удержание объекта в этом состоянии.

3.3. Метод стандартного полинома - биноминальные полином.

**Описание метода**

Задача назначения заданного расположения собственных чисел системы управления (задача модального управления) не эквивалентна задаче максимального быстродействия. Однако возможно приближение к основной задаче за счет поиска соответствующего значения нормирующей частоты выбранного стандартного полинома.

Таким образом, задача модального управления рассматривается как задача поиска такого значения нормирующей частоты выбранного стандартного полинома, которая соответствует минимуму времени переходного процесса по углу рыскания. Эта задача является задачей одномерной оптимизации и может быть решена с помощью MATLAB функции FMINSEARCH.

**Описание кода программы**

Для обеспечения решения задачи FMINSEARCH должна ссылаться на функцию (fminsearch\_function), которая вычисляет время переходного процесса, соответствующее текущему значению нормирующей частоты стандартного полинома (с помощью функции calculate\_transition\_time). Для вычисления времени переходного процесса первоначально вычисляются коэффициенты стандартного полинома, соответствующие текущему значению нормирующей частоты (с помощью функции calculate\_system\_parameters). Затем вычисляются корни полученного полинома, которые рассматриваются в дальнейшем как желаемые корни замкнутой системы управления. Операция вычисления корней может быть выполнена с помощью MATLAB функции ROOTS. Затем по известной математической модели объекта управления и желаемым корням характеристического уравнения замкнутой системы вычисляются параметры линейного алгоритма управления. Последняя операция выполнена с помощью MATLAB функции solve. На последнем этапе с помощью ODE45 вычисляется массив значений угла рыскания и путем обработки массива значений угла рыскания определяется время переходного процесса.

**Код программы**

Файл main.m

|  |
| --- |
| clc; clear; close all;  % Метод 3. Косвенный метод. Стандартный полином.  w0 = 0.3;  [w, time] = fminsearch('fminearch\_function', w0) |

Файл fminsearch\_function.m

|  |
| --- |
| function transitionTime = fminearch\_function(w0)  %@brief This function is used by fminsearch()  %@param w0 - base frequency  %@return transition time  % Step 1. Define global variable and some constans:  global K  x0 = [0 0 10 0];  maxTransitionTime = 40;  % Step 2. Calculate data for system and graph:  calculate\_system\_parameters(w0);  [t, x] = ode45('odefun', [0 maxTransitionTime], x0);  transitionTime = calculate\_transition\_time(t, x);  % Step 3. Show data:  fprintf("w0 = %f, t = %f, k = [%f, %f, %f, %f]\n", w0, transitionTime, K);  % Step 4. Show graphs:  subplot(3, 1, 1); plot(t, x(:, 1:4)); grid on;  legend('омега - угловая скорость рысканья', 'бетта - угол дрейфа', ...  'фи - угол рысканья', 'дельта - угол перекладки руля')  title('ПП угловой скорости рысканья, угла дрейфа, угла рысканья и угла перекладки руля')  subplot(3, 1, 2); plot(t, control\_impact(x')); grid on;  title('Переходные процессы управляющего воздействия')  subplot(3, 1, 3); plot(t, x(:,4)); grid on; title('Переходные процессы угла руля')  pause(0.1)  end  end |

Файл calculate\_system\_parameters.m

|  |
| --- |
| function calculate\_system\_parameters(w0)  %@brief This function calculate main global system parameters (A, B, C)  %@param w0 - base frequency    %Step 1. Define global variables  global A B K    %Step 2. Init data:  % variant 12.  % vessel veriant - 4  % Solution method - standard polynomial p.2.1  V0 = 6.17; L = 39; r21 = -0.69; r31 = 6.14; q21 = 1.22;  q31 = -3.12; s21 = -0.44; s31 = -3.1; W = V0/L; a11 = -r31\*W;  a12 = -q31\*W^2; a21 = -r21; a22 = -q21\*W; b11 = -s31\*W^2; b21 = -s21\*W;    %Step 3. Calculate K, then A and B:  syms k1 k2 k3 k4  characteristicPolynomial = ...  [1; ... %s^4  k4 - a22 - a11; ... %s^3  a11\*a22 - a12\*a21 - a11\*k4 - a22\*k4 + b11\*k1 + b21\*k2; ... %s^2  b11\*k3 - a11\*b21\*k2 + a12\*b21\*k1 + a21\*b11\*k2 - a22\*b11\*k1 + a11\*a22\*k4 - a12\*a21\*k4; ... %s^1  a12\*b21\*k3 - a22\*b11\*k3];    binomialPolynomial = ...  [1\*w0^0; ... %s^4  4\*w0^1; ... %s^3  6\*w0^2; ... %s^2  4\*w0^3; ... %s^1  1\*w0^4]; %s^0    [sk1, sk2, sk3, sk4] = solve(...  characteristicPolynomial(1) == binomialPolynomial(1), ...  characteristicPolynomial(2) == binomialPolynomial(2), ...  characteristicPolynomial(3) == binomialPolynomial(3), ...  characteristicPolynomial(4) == binomialPolynomial(4), ...  characteristicPolynomial(5) == binomialPolynomial(5));  K = [double(sk1), double(sk2), double(sk3), double(sk4)];  A = [a11 a12 0 b11; % угловая скорость рысканья омега\_у  a21 a22 0 b21; % угол дрейфа бетта  1 0 0 0; % угол рысканья фи  -K(1) -K(2) -K(3) -K(4)]; % угол перекладки руля дельта\_в (не больше 35 градусов)  B = [0 0 0 1]; % коэффицинты перед управляющим воздействием  % Step 4. Check roots. All roots should be equal to w0. It's not obligatory  %roots(poly(A))  end |

Файл calculate\_transition\_time.m

|  |
| --- |
| function transitionTime = calculate\_transition\_time(t, x)  %@brief Calculate transition time  %@param t - time vector with size (pointsAmount, 1)  %@param x - state variables matrix with size(pointsAmount, stateVariabelsAmount)  %@return transition time scalar    pointsAmount = length(t);  transitionTime = 0;  initialDeviation = 10;  for i = pointsAmount : -1 : 1  if abs(x(i, 3)) > 0.05\*initialDeviation  transitionTime = t(i);  break;  end  end  end |

Файл control\_impact.m

|  |
| --- |
| function u = control\_impact(x)  %@brief Calculate control impact u = -K1\*x1 - K2\*x2 - K3\*x3 - K4\*x4  %@param x - state variables with size (pointsAmount, stateVariablesAmount)  %@return control impact with size(pointsAmount, 1)  %@note u <= |Umax|  global K  Umax = 1;  u = -K(1).\*x(1, :) - K(2).\*x(2, :) - K(3).\*x(3, :) - K(4).\*x(4, :);  for i = 1:length(u)  if u(i) > Umax  u(i) = Umax;  elseif u(i) < -Umax  u(i) = -Umax;  end  end  end |

Файл odefun.m

|  |
| --- |
| function dxdt = odefun(t, x)  %@brief This function is used by ode45()  %@param t - time vector with size (pointsAmount, 1)  %@param x - state variables matrix with size(pointsAmount, 4)  global A  rudderAngleMax = 35;  if x(4) > rudderAngleMax  x(4) = rudderAngleMax;  elseif x(4) < -rudderAngleMax  x(4) = -rudderAngleMax;  end  dxdt = [A(1,1)\*x(1) + A(1,2)\*x(2) + A(1,3)\*x(3) + A(1,4)\*x(4); ...  A(2,1)\*x(1) + A(2,2)\*x(2) + A(2,3)\*x(3) + A(2,4)\*x(4); ...  A(3,1)\*x(1) + A(3,2)\*x(2) + A(3,3)\*x(3) + A(3,4)\*x(4); ...  control\_impact(x)];  end |

**Результат выполнения программы**

Графики переходных процессов при частоте, соответствующей максимальному быстродействию, представлены на рисунке 4.

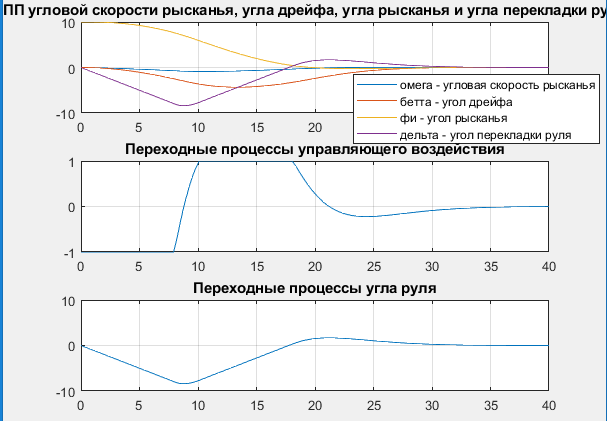


Рисунок 4 – Графики переходных процессов

Таким образом, найдено значение нормирующей частоты выбранного стандартного полинома

w = 0.3572,

которое соответствует минимуму времени переходного процесса по углу рыскания

t = 17.6558

4. Анализ чувствительности реализованных методов

Поскольку задача проектирования решается относительно приближенной математической модели объекта управления, представляется целесообразным проанализировать чувствительность полученных различными методами алгоритмов управления к изменению параметров объекта управления. Эту операцию выполним методом моделирования.

**1.** **Анализ метода, основанного на теореме об N интервалах**

Выполним изменение значения скорости хода относительно исходного значения.

При изменении скорости хода замечается статическая ошибка по углу рысканья, из-за чего невозможно посчитать время переходного процесса.

Значения статической ошибки при разных значениях скорости хода, представлены в таблице 4.

Таблица 4. Статическая ошибка при разных значениях скорости хода.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Значение скорости хода, V | 0.7V0 | 0.8V0 | 0.9V0 | V0 | 1.1V0 | 1.2V0 | 1.3V0 |
| Значение статической ошибки | 3.542 | 2.373 | 1.197 | 0 | -1.204 | -2.428 | -3.608 |

Графики переходного процесса при значениях скорости хода 0.7\*V0 и 1.3\*V0 представлены соответственно на рисунках 5 и 6.

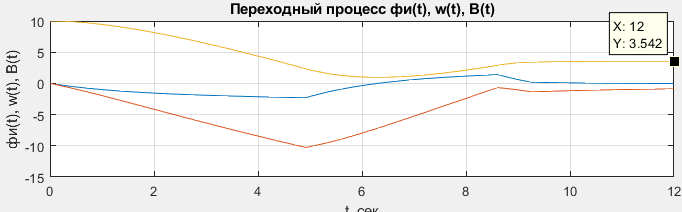


Рисунок 5 – Скорость хода 70% в методе, основанном на N интервалов

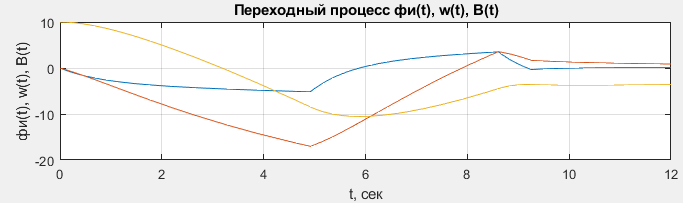


Рисунок 6 – Скорость хода 130% в методе, основанном на N интервалов

При изменении длины по ватерлинии так же появляется ошибка, но ее значение обратно значению статической ошибки скорости хода.

Проводить опыты с изменением двух параметров одновременно нецелесообразно, поскольку значение нормирующей частоты определяется как

.

Таким образом, даже при небольших отклонениях от исходных параметров, алгоритм дает значительные погрешности.

**2. Анализ метода параметрической оптимизации**

Выполним изменение значения скорости хода относительно исходного значения. Исходное время переходного процесса по углу рысканья 17.75 сек.

При изменении скорости хода, меняется время переходного процесса. Значения времени переходного процесса от скорости хода представлены в таблице 5.

Таблица 5.Время переходного процесса при разных значениях скорости хода.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Значение скорости хода, V | 0.7V0 | 0.8V0 | 0.9V0 | V0 | 1.1V0 | 1.2V0 | 1.3V0 |
| Время переходного процесса | 36.55 | 31.4 | 26.7 | 17.75 | 17.7 | 22.8 | 24.46 |

Графики переходного процесса при значениях скорости хода 0.7\*V0 и 1.3\*V0 представлены соответственно на рисунках 7 и 8.

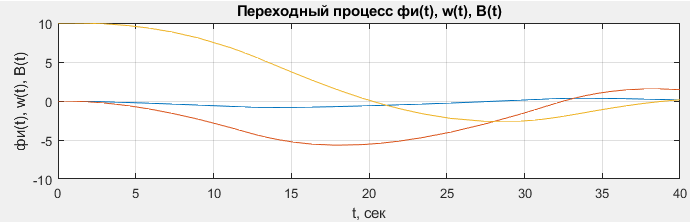


Рисунок 7 – Скорость хода 70% в методе стандартного полинома

При увеличении скорости хода, время переходного процесса по углу рысканья уменьшается на 5% до 16.6 сек. Полученный график представлен на рисунке 8.

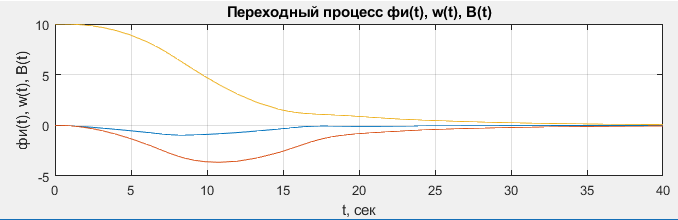


Рисунок 8 – Скорость хода 130% в методе стандартного полинома

При изменении длины по ватерлинии время переходного процесса изменяется схожим образом, но его значение обратно значению скорости хода.

Таким образом, при изменении параметров объекта, увеличивается значение переходного процесса.

**Анализ метода стандартного полинома – биномиальный полином**

Выполним изменение значения скорости хода относительно исходного значения. Исходное время переходного процесса по углу рысканья 17.65 сек.

При изменении скорости хода, меняется время переходного процесса. Значения времени переходного процесса от скорости хода представлены в таблице 6. Графики процессов представлены на рисунках 9 и 10.

Таблица 6.Время переходного процесса при разных значениях скорости хода.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Значение скорости хода, V | 0.7V0 | 0.8V0 | 0.9V0 | V0 | 1.1V0 | 1.2V0 | 1.3V0 |
| Время переходного процесса | 32.51 | 27.7 | 18.51 | 17.65 | 17.28 | 16.83 | 16.69 |

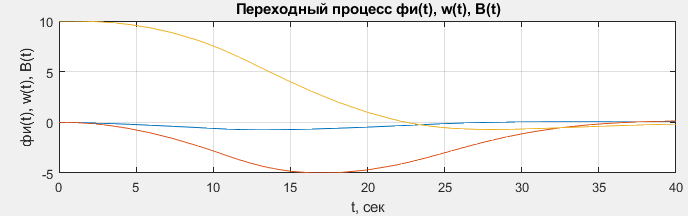


Рисунок 9 – Скорость хода 70% в методе параметрической оптимизации

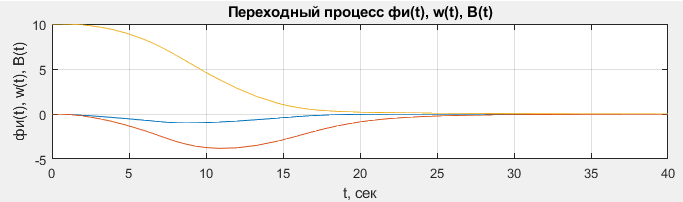


Рисунок 10 – Скорость хода 130% в методе параметрической оптимизации

При изменении длины по ватерлинии время переходного процесса изменяется схожим образом, но его значение обратно значению скорости хода.

Таким образом, чем больше скорость хода, тем быстрее переходный процесс.

Заключение

В ходе работы был спроектирован алгоритм управления динамическим объектом на примере водоизмещающего судна.

Проектирование алгоритма управления было выполнено тремя методами:

1. метод, основанный на теореме об N интервалах

2. метод параметрической оптимизации линейного закона управления

3. метод стандартного полинома – биномиальный полином

Спроектированный алгоритм управления рулем судна обеспечивает минимальное время устранения начального значения угла рыскания равного 10°.

Методом, основанным на теореме об N интервалов, удалось получить время переходного процесса 9.1 сек. Методы параметрической оптимизации и стандартного полинома дали соответственно 16.75 и 17.66 сек.

Таким образом, первый метод достигает максимального быстродействия.

Однако, в результате анализа чувствительности полученных различными методами алгоритмов управления к изменению параметров объекта управления, было замечено следующее:

- при небольших изменениях параметров, первый метод приобретает статическую ошибку – вычислить время переходного процесса в таком случае невозможно, поскольку установившееся значение угла рысканья может превышать 5% от начального значения;

- метод параметрической оптимизации очень сильно увеличивает время переходного процесса по углу рысканья, при уменьшении скорости хода;

- метод стандартного полинома имеет наименьшую чувствительность к изменению параметров системы.

Сложность задач, решаемых на этапе проектирования, и реализации алгоритма управления не большая.

Список литературы

1. Методические указания к курсовому расчету «ПРОЕКТИРОВАНИЕ ОПТИМАЛЬНЫХ СИСТЕМ УПРАВЛЕНИЯ», Санкт–Петербург, 2013

2. Методические указания к лабораторным работам «ПРОЕКТИРОВАНИЕ ОПТИМАЛЬНЫХ СИСТЕМ УПРАВЛЕНИЯ», В. А. Зуев А. С. Ветчинкин, Санкт–Петербург, 2013

3. Конспект лекций по дисциплине «Проектирование оптимальных систем управления»